日本国特許庁
JAPAN PATENT OFFICE

24.12.03

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００２年１０月２１日

出願番号
Application Number: 特願２００２－３０６２９６

[ST. 10/C]: [ＪＰ２００２－３０６２９６]

出願人
Applicant(s): アネックスシステムズ株式会社
玉津　雅晴

２００４年　２月　５日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

出証番号　出証特２００４－３００６２６０

【書類名】　特許願
【整理番号】　021-035
【あて先】　特許庁長官殿
【国際特許分類】　G06F 12/00
G02F 17/30
【発明の名称】　データおよびデータベースのアクセラレーター・システム
【請求項の数】　2
【発明者】
【住所又は居所】　東京都多摩市馬引沢２丁目１４番１４号　サンセットヒルズ２
【氏名】　玉津雅晴
【特許出願人】
【識別番号】　593050596
【氏名又は名称】　アネックスシステムズ株式会社
【特許出願人】
【識別番号】　592159324
【氏名又は名称】　玉津雅晴
【代理人】
【識別番号】　100082418
【弁理士】
【氏名又は名称】　山口朔生
【選任した代理人】
【識別番号】　100099450
【弁理士】
【氏名又は名称】　河西祐一

【選任した代理人】

【識別番号】　100114867

【弁理士】

【氏名又は名称】　横山正治

【手数料の表示】

【予納台帳番号】　033569

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　データおよびデータベースのアクセラレーター・システム

【特許請求の範囲】

【請求項１】　一つのユニークな主キーとゼロ個または１個以上のノン・ユニークな代替キーを持つレコードを順次格納するブロックと、前記ブロックの位置管理を、当該ブロックとランダム・アクセス・メモリ上のアドレスとを対応させたロケーション・テーブルを用いて行い、前記ランダム・アクセス・メモリに格納されているデータ及びデータベースを管理するコンピューターによるシステムであって、

前記、プライマリー・システムに対して、ロケーション・テーブルを保有し、ブロックを保有しないアクセラレーター・システムを１つ以上使用し、

前記プライマリー・システムのロケーション・テーブルに変更がなされたときは、その変更をアクセラレーター・システムに通信する手段を有し、

前記、変更をアクセラレーター・システムのロケーション・テーブルに同期させて更新させる手段を有する、

ことを特徴とする、アクセラレーター・システム。

【請求項２】　一つのユニークな主キーとゼロ個または１個以上のノン・ユニークな代替キーを持つレコードを順次格納するブロックと、前記ブロックの位置管理を、当該ブロックとランダム・アクセス・メモリ上のアドレスとを対応させたロケーション・テーブルを用いて行い、前記ランダム・アクセス・メモリに格納されているデータ及びデータベースを管理するコンピューターによるシステムであって、

代替キーによる検索・更新を可能とするため、代替キーと当該代替キー値のレコードが格納されているブロック番号と当該レコードの主キーとからなるエントリーを代替キーの順番に並べて複数格納可能とするとともに、予め同一の大きさで必要数を連続して確保できる代替キー・ブロックを使用し、代替キー・テーブルのエントリーは代替キー・ブロック中に代替キーの順番で並ぶように格納されており、同一の代替キーを持つ代替キー・テーブルのエントリーは同一の代替キー・ブロックに格納されており、同一の代替キーを持つエントリーの数が多いか

もしくは代替キーの挿入により、代替キー・ブロックに格納できないときに、代替キー・オーバーフロー・ブロックを代替キー・ブロックに追加してエントリーを格納し、その代替キー・ブロックとランダム・アクセス・メモリ上のアドレスとを対応させた代替キー・ロケーション・テーブルを用いて行い、前記ランダム・アクセス・メモリに格納されているデータ及びデータベースを管理するコンピューターによるシステムであって

前記プライマリー・システムに対して、代替キー・ロケーション・テーブルを保有し、代替キー・ブロックを保有しないアクセラレーター・システムを１つ以上使用し、

前記プライマリー・システムの代替キー・ロケーション・テーブルに変更がなされたときは、その変更をアクセラレーター・システムに通信する手段を有し、

前記、変更をアクセラレーター・システムの代替キー・ロケーション・テーブルに同期させて更新させる手段を有する、

ことを特徴とする、アクセラレーター・システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はコンピューターにおけるデータ格納・検索方式及びシステムに係り、特にデータ格納および検索に対して、大きなスケーラビリティ（処理能力の拡張性）を与えることを目的とするものである。スケーラビリティとは、処理能力の拡張性のことである。単一のデータベース・システムが、ある一定の性能を有している場合に、その性能に対して、数倍から数千倍以上の処理能力を提供するもので、データベース・システムの性能向上に関するものである。

【０００２】

【従来の技術】

従来のコンピューターによるデータベース格納検索方式は、「Jeffrey D.Ullman著、国井他１名訳、”データベース・システムの原理”，第１版，日本コンピューター協会，１９８５年５月２５日，ｐ４５－７１」、「Samuel Leffler他著／

中村明他訳，”ＵＮＩＸ４．３ＢＳＤの設計と実装”，丸善株式会社，１９９１年６月３０日，ｐ１９３－１９１」及び「Michael J.Folk他著／楠本博之訳，”ｂｉｔ別冊　ファイル構造”，共立出版　株式会社，１９９７年６月５日，ｐ１６９－１９１」に記載されているように、基本的には階層型のインデックスを使用するものであった。

【０００３】

このような従来のデータベース格納検索方式によれば、

（１）インデックスの創生や維持に負荷がかかること、

（２）最終的に使用されると想定される最大のブロックを予め生成しておかなければならないこと、

（３）インデックスが階層構造をとっているため、データの挿入や削除によってインデックスの更新が行われる場合に、上位インデックスまで変更される場合が発生するために排他範囲が広くなり、デッドロックを引き起こしやすいこと、などの不都合があった。

【０００４】

このような従来のデータベース格納検索方式の不都合を解消するため、本発明者は、従来の階層型インデックスに代えてロケーション・テーブルと代替キー・テーブルという概念を導入し、インデックスの処理に伴う複雑な処理を簡素化し、テーブル自体の検索をバイナリー・サーチなどの手法を用いることにより、高速化と、メンテナンスの容易性を確保できるようにした「データ格納検索方式」を提案した（日本国特許第３３４５６２８、米国特許第６４１５３７５号参照）。

【０００５】

ここで、本発明者が提案したデータ格納検索方式の内容について簡単に説明しておくことにする。本発明のデータ格納検索方式は、ロケーション・テーブルと代替キー・テーブルを使用し、それらに対して、バイナリー・サーチを行うことにより、目的のレコードを検索するものである。レコードはブロックと言う固定長の格納領域に格納する。レコードは、当初はプライマリー・ブロックに格納するが、そのプライマリー・ブロックが満杯になった後、そのプライマリー・ブロ

ックに対して、挿入レコードが発生するとレコードを格納できなくなるため、当該プライマリー・ブロックに対して、オーバーフロー・ブロックを作成し連結し、一部のレコードをオーバーフロー・ブロックに移動した後、プライマリー・ブロックにレコードの格納を行う。そのオーバーフロー・ブロックが満杯になった後で、更にレコードの挿入が発生すると、そのオーバーフロー・ブロックに対して、オーバーフロー・ブロックを作成して連結する。

【０００６】

ここで、連結とは、物理的に連結されていることを意味するのではなく、プライマリー・ブロックが一番目のオーバーフロー・ブロックのアドレスを保持し、１番目のオーバーフロー・ブロックが２番目のオーバーフロー・ブロックのアドレスを保持している状態が、あたかも物理的にブロックが繋がれているように扱えることから、そのような表現を使用している（以下、同様である）。

【０００７】

ロケーション・テーブルは予め連続領域に必要な大きさを確保しておき、プライマリー・ブロックの管理を行う。オーバーフロー・ブロックは、上記のようにプライマリー・ブロックに連結されている。

【０００８】

代替キーによる検索を可能とするために、代替キー・テーブルを作成する。代替キー・テーブルは、固定長の代替キー・ブロックを連続領域に確保し、そのブロック中に、代替キー・エントリーを代替キー値の順番に格納する。代替キー・ブロックが満杯になって代替キー・エントリーが格納できなくなったら、代替キー・オーバーフロー・ブロックを連結する。バイナリー・サーチは代替キー・ブロックに対して行う。代替キー・テーブルは代替キーの種類に応じて何種類でも作成できる。

【０００９】

更に、特願２００２－２６４２８３号「データ及びデータベースの無停止再編成システム並びにデータ及びデータベース」では、代替キー・ブロックに対して、代替キー・ロケーション・テーブルを付加し、再編成を容易にする発明を行っている。

【００１０】

このように本発明者の提案した「データ格納検索方式」では、オーバーフロー・ブロックは無制限に連結できるので、レコードが格納できなくなる状態は発生しないという利点がある。

【００１１】

以上のような特徴を持つ「データ格納検索方式」では、ハードウエアの性能を最大限に引き出し、大幅な高速化を実現することが可能であるが、一方で、その性能の上限になると、それ以上の処理性能を提供することができない、つまり、スケーラビリティが無い方式であった。これは、従来の他の方法も本質的には同様である。但し、従来の方法はハードディスクの使用を前提に考えられているため、これを、半導体などの高速メモリー上に乗せることにより、ハードディスク上で実現しているスピードに対して、高速化を図ることが可能であったが、高速メモリーの性能限界が上限になるものであり、本質的なスケーラビリティを提供するものではなかった。

【００１２】

従来の技術でスケーラビリティを向上させる技術とされるものに、ロード・バランサーが上げられる。これは、サーバーを複数用意して、それが見かけ上一台のサーバーであるかのようにするものである。外部からの処理要求が過大な場合、まず、ロード・バランサーに処理要求を入れ、ロード・バランサーが処理要求を複数のサーバーの何れかに割り当てることにより、一台当りの処理負荷を低減させ、それによりスケーラビリティの向上を図ろうとするものである。しかしながら、この方式には致命的な欠陥があった。それは、複数のサーバーが処理できるのは、処理のロジック部分のみであり、データベースは一つであり、各サーバーから同一のデータベースをアクセスするため、データベースの性能が処理性能を規定してしまうということであった。つまり、処理要求が増加して、サーバーを増加させても、データベースの性能が上限となってしまうということである。

【００１３】

「データ格納検索方式」においてスケーラビリティを確保する方法の一つとして、国内優先「特願２００１－０９４６７８　データバックアップ・リカバリー

方式」で、実際のデータ更新・参照用のプライマリー・システムに対して、そのバックアップ・コピーであるセカンダリー・システムを設け、本来的な目的としてはバックアップ・リカバリー用のシステムとして利用する他に、セカンダリー・システムを参照用のシステムとして使用することにより、一定のスケーラビリティが確保できることが考案されている。

【００１４】

この方法では、更新トランザクション量が、参照トランザクション量に比較して、一般的に１対１０程度であることから、セカンダリー・システムを参照用に使用することにより、プライマリー・システムの負荷を軽減することが可能であるとするものである。しかしながら、セカンダリー・システムを更新用には使用できないことから、スケーラビリティの確保としては限界が低いものであった。また、この方法では、セカンダリー・システムは、プライマリー・システムと同等の構成が必要であり、ロケーション・テーブル、データ格納用ファイル（ブロックの集合）、代替キー・テーブル（ゼロ、または、１つ、または、複数）をセットとして用意するものであったため、バックアップの目的以外にスケーラビリティ確保の目的として、セカンダリー・システムを複数用意することは、費用的に負担が大きくなるものであった。

【００１５】

「データ格納検索方式」以外の従来の方法では、上記以外に、データを保持するサーバーのミラー化によって性能向上を図ることが一般的に行われているが、ミラーとなるデータベースは、元のデータベースと同じ容量が必要であるため、大きな格納容量を必要とするものである上に、ミラー・データベースは基本的に参照用にしか使用できず、更新は制限が大きなものであった。主サーバーのデータが変更された場合に、ミラー・サーバーに、その更新内容を反映することが、一定時間遅れてしか実現できないという制限である。これは、ミラー・サーバーでデータの更新を行うと、主サーバーでの更新との時間差（タイムラグ）のため、更新結果が無効になってしまう危険性があるということであり、通常のデータ処理では採用できない方法であった。このように、ミラー・サーバーを使用した性能向上は、通常のリアル・タイム・データ処理では使用できないものであった

。

【００１６】

このように、データベースに対するスケーラビリティの確保は重要な課題であるが、十分な解決方法が存在してこなかったのが実情である。これは、従来の方法では、データそのものを複写する必要があるのに加え、インデックス構造が複雑なために、インデックスが更新された場合の複写が面倒であった、ということに起因している。

【００１７】

また、本発明に関連する既存の発明として、国内優先「特願２００１－０９４６７８　データバックアップ・リカバリー方式」と、先に挙げた「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」が挙げられる。

【００１８】

【発明が解決しようとする課題】

スケーラビリティは、データ格納・検索方式（本明細書では、データベースまたはデータベース・システムと呼ぶこともある）を実際に使用する際には、重要な問題となる。システムを構築する際に、トランザクション（処理要求）量が比較的一定している場合には大きな問題とはならないが、トランザクション量が増大していく場合には、システム性能の拡張性が問題となる。それは、データベースに要求される性能が限界に達してしまうと、それ以上の処理が出来ないばかりか、トランザクションが溜まる一方になってしまうため、業務が停止してしまうからである。処理能力の限界が現用業務量に対して遥かに高い場合は問題ないが、それが接近している（通常は２から３倍以内）場合には、業務の拡大に伴って現用業務が停止する危険性があるため、採用を躊躇うことに繋がってしまう。処理量が増加しても、それに追随してスケーラビリティが確保できるデータベースは、情報処理において重要な要求であった。

【００１９】

【課題を解決する為の手段】

「データ検索格納方式」を用いてデータベース・システムを構築し、主キー（

異なるレコードのキー値が重複しないユニークなキーで、レコードに一つ存在するもの）での検索の場合を考えてみる。主キーの検索は、ロケーション・テーブルに対してバイナリー・サーチを用いて行うが、この場合のアクセス回数は、以下のようになる。

【００２０】

仮に、データ件数が、１０億件ある場合で、ブロックに平均して２０レコード格納できるとすると、ブロックの数は５０００万となる。オーバーフロー・ブロックが発生していない場合、５０００万ブロックに対してバイナリー・サーチを行って対象となるブロックを検索することになるが、バイナリー・サーチの回数は、底を２とするｌｏｇの値となる。この場合は、２５．６となり、切り上げて２６回が解となる。

【００２１】

一方、ブロック内におけるレコードの検索は、特殊な方法を使わずに、先頭からレコードを読んでいった場合で、期待値は１０回となる。

つまり、一つのブロックを検出するために、ロケーション・テーブルを２６回検索する必要があることを示している。言い換えれば、ロケーション・テーブルに対するアクセスの負荷が、ブロックに対する負荷よりも高いことを示している。

【００２２】

このことは、代替キー（異なるレコードのキー値が重複してもかまわないノン・ユニークなキーで、レコード中に複数種類が存在しても良い）を用いた検索でも同様である。代替キーを用いた検索は、代替キー・テーブルに対してバイナリー・サーチを行うことによって行う。代替キー・テーブルの一つのブロックが平均的に５０のエントリーを保持しているとする。代替キー・テーブルのブロック数は２０００万ブロックとなる。２０００万ブロックに対する、バイナリー・サーチの回数は、２４．３となり、切り上げて２５回となる。

代替キー・ブロック内の代替キー・エントリーは固定長とすることができるので、代替キー・エントリーをバイナリー・サーチを用いて検索する場合は、６回となる。

【００２３】

つまり、ロケーション・テーブルと同程度の負荷が代替キー・テーブルに存在することになり、代替キー・テーブルに対する負荷が、代替キー・ブロックに対する負荷を大きく上回ることになる。

【００２４】

また、同一のブロックをアクセスする確率は、かなり低いものである。何故ならば、トランザクションの平均時間が１００分の１秒として、１秒あたりのトランザクション処理数が１万件だとし、トランザクションあたりのデータ更新件数が５０件とすると、以下のようになる。

【００２５】

あるトランザクションが実行中に、他のトランザクションが実行されるのは、平均的に１００トランザクションとなる。となると、その時点で同時に更新が行われるレコードは、１００×５０＝５０００となる。

【００２６】

一方、ブロックの総数は５０００万ブロックであるから、ブロック排他を前提とした場合でも、同一のブロックに格納されているレコードが同時に使用される確率は、非常に低いものとなる。

【００２７】

このことは、ロケーション・テーブルや代替キー・テーブルを複数用意し、最新のエントリーをコピーして、キー検索の可能パス数を増加し、更に、各パスに対してＣＰＵを割り当てることにより、処理データ量を増加することができることを示している。

【００２８】

ここで、データベースを保持して検索・更新を行うサーバーをプライマリー・システムと呼ぶことにする。図１はプライマリー・システムの一例である。データを格納するブロックと、ブロックを管理するロケーション・テーブルの組、代替キー・エントリーを格納する代替キー・ブロックと、代替キー・ブロックを管理する代替キー・ロケーション・テーブルの組（代替キー・テーブル）とからなる。代替キー・テーブルは、代替キーの種類毎に作成する。代替キーの種類とは、社員マスターにおける、所属や生年月日、入社年月日などである。

【００２９】

図１では、代替キーの種類が３種類の場合を表している。ここでは、本発明者が発明した「データ格納検索方式」で提案した、代替キー・テーブルに代えて、特願２００２－２６４２８３号「データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で発明した、代替キー・テーブルに代替キー・ロケーション・テーブルを使用する方式を採用している。

【００３０】

図２は、プライマリー・システムとアクセラレーター・システムの例である。アクセラレーター・システムとは、プライマリー・システムの機能の内、ロケーション・テーブルと代替キー・ロケーション・テーブルを保持して、それらのテーブルをプライマリー・システムと同期させる機能と、それらのテーブルを用いてレコードが格納されているブロックを検索する機能を保持している。

【００３１】

図１のプライマリー・システムで、代替キー・テーブルに対して代替キー・ロケーション・テーブルを保持する理由は、代替キー・ロケーション・テーブルを使用しない場合には、代替キー・ブロックの変更を、プライマリー・システムからアクセラレーター・システムに送信する必要があるのに比べて、代替キー・ロケーション・テーブルを使用した場合には、代替キー・ブロックが変更になっても、代替キー・ロケーション・テーブルに変更が無ければ、プライマリー・システムからアクセラレーター・システムに変更を送信する必要が無く、代替キー・ロケーション・テーブルの変更は少ないので、プライマリー・システムからアクセラレーター・システムに対する、データ送信量を少なくすることが可能であるからである。

【００３２】

プライマリー・システムで、データの追加・更新・削除などにより、ブロック内のデータが書き換えられ、それに伴ってロケーション・テーブルが変更された場合には、プライマリー・システムからアクセラレーター・システムに対して、ロケーション・テーブルの変更箇所（エントリーの番号）と変更内容を送信して、アクセラレーター・システムの当該エントリーの更新を行う。

【００３３】

同様に、プライマリー・システムで、データの追加・更新・削除などにより、ブロック内のデータが書き換えられ、それに伴って代替キーが変更された場合、代替キー・ブロック内のエントリーの書き換えが行われるが、それに伴って代替キー・ロケーション・テーブルの更新が行われた場合には、プライマリー・システムからアクセラレーター・システムに対して、代替キー・ロケーション・テーブルの変更箇所（エントリーの番号）と変更内容を送信して、アクセラレーター・システムの当該エントリーの更新を行う。

このようにして、アクセラレーター・システムでは、最新のロケーション・テーブル及び代替キー・ロケーション・テーブルを保持する。

【００３４】

図３及び図４は、本発明を実際の情報処理に適用する場合の、処理サーバー（アプリケーション・サーバー）とデータベース・サーバーの関係を示したものである。図３では、ロード・バランサーを用いて、外部からの処理要求をアプリケーション・サーバーに振り分けている場合である。図４は、処理要求振り分けシステムを作成してアプリケーション・サーバーとデータベース・サーバーを接続する場合である。

【００３５】

図３の場合、外部からの処理要求は、ロード・バランサーによって、アプリケーション・サーバーに割り当てられる。従来の方法では、各アプリケーション・サーバーの右側には、データベース・サーバーが一つ接続されることになるが、本発明の場合には、各アプリケーション・サーバーに対して、プライマリー・システムかアクセラレーター・システムが接続されることになる。

【００３６】

処理サーバー０に対する処理要求に基づいてデータの検索を行う場合には、プライマリー・システムに対して要求を出す。プライマリー・システムでは、その要求が主キーである場合にはロケーション・テーブルを使用して目的のレコードを検索する。代替キーである場合には、代替キー・ロケーション・テーブルを使用して代替キー・エントリーを検索し、その代替キー・エントリーに基づいてロ

ケーション・テーブルを検索して、目的のレコードを検索する。

【００３７】

処理サーバー１に対する処理要求に基づいてデータの検索を行う場合には、アクセラレーター・システム１に対して要求を出す。アクセラレーター・システムでは、その要求が主キーである場合にはアクセラレーター・システム１のロケーション・テーブルを使用して目的のロケーション・テーブル・エントリーを探し出す。ロケーション・テーブル・エントリーには、対象ブロックのアドレスが記述してあるが、アクセラレーター・システム１にはブロックが無い。これは、プライマリー・システムのブロック及びブロック内のレコードを参照する。

【００３８】

検索が代替キーである場合には、アクセラレーター・システム１の代替キー・ロケーション・テーブルを使用して代替キー・エントリーを検索し、その代替キー・エントリーに基づいてアクセラレーター・システム１のロケーション・テーブルを検索して、目的のロケーション・テーブル・エントリーを検索する。その次には、プライマリー・システムのブロック及びブロック内のレコードを参照する。

以上が、アクセラレーター・システム１の場合に関する説明であるが、アクセラレーター・システム２以下もアクセラレーター・システム１の場合と全く同様である。

【００３９】

前記例では、データの検索の場合に関して記述したが、これは、レコードの追加・更新・削除の場合も全く同様である。アクセラレーター・システム１に対するデータ処理要求に基づいてアクセラレーター・システム１内で、該当ロケーション・テーブル・エントリーの検索までを行い、その後、プライマリー・システムのブロック及びブロック内のレコードの検索を行い、更に更新・追加・削除を行う。

【００４０】

以上のようにアクセラレーター・システムを設けて、ロケーション・テーブルと代替キー・ロケーション・テーブルの複製を用意することにより、データベー

スに対する負荷を分散することが可能となる。また、アクセラレーター・システムは、ブロック及び代替キー・ブロックを保持しないため、データ格納容量としては、プライマリー・システムに比較して、数十分の１から数百分の１で済み、システムを拡張する場合に必要な費用が非常に少なくて済むことになる。

【００４１】

【実施例】

本発明は、（日本国特許第３３４５６２８号、米国特許第６４１５３７５号参照）「データ格納検索方式」の発想を踏まえて、基本部分をそのまま利用しながら、スケーラビリティの確保を図ることを目的としている。

【００４２】

本発明では、スケーラビリティを確保する方法として、「データ格納検索方式」で示された、ロケーション・テーブルを複数用意し、更に「特願２００２－２６４２８３号データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で示された代替キー・ロケーション・テーブルを複数用意するというものである。

【００４３】

従来の方法のスケーラビリティを確保する方法の一つとして、国内優先「特願２００１－０９４６７８　データバックアップ・リカバリー方式」で、実際のデータ更新・参照用のプライマリー・システムに対して、そのバックアップ・コピーであるセカンダリー・システムを設け、本来的な目的としてはバックアップ・リカバリー用のシステムとして利用する他に、セカンダリー・システムを参照用のシステムとして使用することにより、一定のスケーラビリティが確保できることが考案されている。しかしながら、参照用としての利用に限定されていた。

【００４４】

以下に、「データ格納検索方式」に関して説明を行う。「データ格納検索方式」の特徴の第１は、ロケーション・テーブルという、レコードを格納するためのブロック（プライマリー・ブロック及びオーバーフロー・ブロック）を、ロケーション・テーブルというフラットな（階層構造を持たない）テーブルを用いて管理している点である。ブロック間では、あるブロックより前にあるブロックに格

納されているレコードの主キー値は、あるブロックに格納されているレコードの主キー値より小さい。ブロックには、レコードをそのレコードの主キーの順番に格納するようにする。これは、プライマリー・ブロック内でも、プライマリー・ブロックに連結されているオーバーフロー・ブロック内でも、また、その相互間でも同様である。このようにすることで、ブロック内のレコードを検索する際の効率が向上する。

【００４５】

「主キー」とは、一つの種類のデータ上でユニークなキーであり、レコードに一つ必要となるものである。例えば、従業員マスターにおける、従業員コード、取引先マスターにおける取引先コードなどである。

【００４６】

「代替キー」とは、データ上でノン・ユニークなキーであり、レコード中に複数種類の代替キーが存在してもよい。例えば、従業員マスターにおける、従業員氏名や所属、入社年月日などである。

【００４７】

ブロック内にレコードの挿入がある場合、そのブロックに格納できなくなった場合には、そのブロックに対して、オーバーフロー・ブロックを追加し、双方を連続して使用することにより、レコードの格納を行う。オーバーフロー・ブロックが一つで足りなくなった場合には、そのオーバーフロー・ブロックに対して更にオーバーフロー・ブロックを連結し、順次オーバーフロー・ブロックを連結することにより、レコード挿入が幾つでも可能としてある。

【００４８】

ロケーション・テーブルはプライマリー・ブロックの管理のみを行い、オーバーフロー・ブロックはプライマリー・ブロックの従属として、ロケーション・テーブルでは管理されない。このため、オーバーフロー・ブロックが発生しても、ロケーション・テーブルの構造的な変更は発生しない。

【００４９】

上記データ格納検索方式の特徴の第２は、主キーによる検索には、ロケーション・テーブルをバイナリー・サーチして目的のブロックを求めるという方法によ

り、旧来のインデックスを使用せず、インデックス管理を効率化し、検索・格納を高速化した点である。

【００５０】

上記データ格納検索方式の特徴の第３は、代替キー用に代替キー・テーブルという、これもフラットなテーブルを採用する。「代替キー」とは、既に説明してあるが、再度繰り返すと、データ上でノン・ユニークなキーであり、レコード中に複数種類の代替キーが存在してもよい。例えば、従業員マスターにおける、従業員氏名や所属、入社年月日などである。

【００５１】

代替キーの追加や変更によりキー値のエントリーが増加しても、代替キー・テーブル・ブロックに代替キー・オーバーフロー・ブロックを追加することで、旧来のインデックスにあったような分割をなくし、代替キー用のインデックス管理を効率化したことである。また、代替キー・オーバーフロー・ブロックが一つで不足する場合には、その代替キー・オーバーフロー・ブロックに対して、更に、代替キー・オーバーフロー・ブロックを連結することにより、代替キー・エントリーの挿入に関する制限を無くしている。

【００５２】

上記データ格納検索方式の特徴の第４は、代替キーによる検索について、代替キー・テーブルをバイナリー・サーチにより検索することにより高速化したものである。

【００５３】

更に、上記データ格納検索方式の特徴の第５は、代替キー・テーブルに対するキー値の追加変更が多く行われると、代替キー・オーバーフロー・ブロックが増加するため、検索効率が低下する可能性があるが、プレ代替キー・テーブルを使用することにより、検索効率が低下しないような工夫がされている。

【００５４】

更に「特願２００２－２６４２８３号データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」では、「データ格納検索方式」で示された、代替キー・テーブルの形式に関して、新たな形式を追加している

。「データ格納検索方式」では、代替キー・ブロックというブロックに代替キー・エントリーを代替キーの順番に格納し、代替キー・ブロックに対してバイナリー・サーチを行う、というものであった。一方、「データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」では、代替キー・ブロックに対して、代替キー・ロケーション・テーブルを作成して、代替キー・ブロックの管理を行う、というもので、代替キーの検索も代替キー・ロケーション・テーブルに対して実行する。

【００５５】

ここで、ファイル（ブロックの集合）とロケーション・テーブル、代替キー・テーブル（種類ごとに１つの代替キー・ロケーション・テーブルと代替キー・ブロック）の組合せをプライマリー・システムと呼ぶ。また、プライマリー・システム以外のロケーション・テーブル、代替キー・ロケーション・テーブルをアクセラレーター（加速器）またはアクセラレーター・システム（加速システム）と呼ぶことにする。

【００５６】

まず、プライマリー・システム１つについて、アクセラレーター・システム１つという構成に関して述べる。

図２はアクセラレーター・システムの構成例である。ロケーション・テーブルＬ１と代替キー・テーブルＡ１、Ｂ１、Ｃ１はプライマリー・システムとアクセラレーター・システムとで、同一の構成・構造である。つまり、プライマリー・システムのロケーション・テーブルＬ０に対して、アクセラレーター・システムのロケーション・テーブルＬ１が割り当てられる。同様に、プライマリー・システムの代替キー・テーブルＡ０に対して、アクセラレーター・システムの代替キー・テーブルＡ１が割り当てられる。更に同様に、プライマリー・システムの代替キー・テーブルＢ０に対して、アクセラレーター・システムの代替キー・テーブルＢ１が割り当てられる。同様に、Ｃ０に対してＣ１が割り当てられる。

ここで、Ａ０とＡ１、Ｂ０とＢ１、Ｃ０とＣ１は構成・構造が同一である。

【００５７】

「データバックアップ・リカバリー方式」では、プライマリー・システムに格

納されているデータに対する変更を、セカンダリー・システムに反映させる方式として、同期密結合方式と非同期疎結合方式を提案してあるが、本発明では、この内同期密結合方式を用いる。

【００５８】

［ロケーション・テーブルの更新をアクセラレーター・システムに反映させる方式］

まず、プライマリー・システム０におけるロケーション・テーブルＬ０の更新を、アクセラレーター・システム１に反映させる方式に関して述べる。ロケーション・テーブルＬ０が更新される、ということは、ロケーション・テーブルＬ０の或るエントリーの内容が更新される、ということである。この場合、プライマリー・システム０のロケーション・テーブルＬ０のエントリー「ｎ」の内容が更新された場合、ロケーション・テーブルＬ０のエントリー番号「ｎ」と更新後の内容を、アクセラレーター・システム１に送信する。送信は、既述したように同期密結合方式を用いて行う。また、トランザクションの種類としては、Ａログ（更新後ログ）となる。また、どのファイルに対する更新かを示すために、ロケーション・テーブルＬ０の識別を付する。この識別も、国内優先「特願２００１－０９４６７８　データバックアップ・リカバリー方式」で述べられているものである。

【００５９】

更新は、内容の更新のほかに、エントリーの追加がある。エントリーは、「データ格納検索方式」を使用する際に、最初に、それまでに存在するレコードを格納したブロックの分に関しては作成しておく。既に作成されているロケーション・テーブルＬ０に対して、追加エントリーが作成される場合について記述する。

【００６０】

この場合、エントリーの「ｍ」まで使用しており、「ｍ＋１」を新たに使用する、ということになる。そうすると、エントリー「ｍ＋１」の内容に関しては、更新の場合と全く同一であることになる。つまり、プライマリー・システム０のロケーション・テーブルＬ０のエントリー「ｍ＋１」の更新後の内容を、エントリーの番号と共に、アクセラレーター・システム１に送信する。但し、この場合

、最終ポインターも更新されているので、最終ポインターの内容も、エントリーと同様に、アクセラレーター・システム１に送信する必要がある。

【００６１】

次に、予め作成しておいたロケーション・テーブルを総て使用してしまい、全く新たにロケーション・テーブルを作成し、そのロケーション・テーブルに対してエントリーを作成する場合に関して、記述する。

【００６２】

この場合は、最初に作成したロケーション・テーブルのエントリーの数がｋ個だとする。ロケーション・テーブルの追加はエントリー１つ分づつ行うことも可能であるが、効率が良くないため、ある程度まとまったロケーション・テーブルを、連続領域に作成することが好ましい。そこで、「ｋ＋１」から「ｋ＋ｈ」まで、ｈ個のエントリーを持つロケーション・テーブルを追加する場合に関して記述する。

【００６３】

この場合、まず、連続領域にｈ個のエントリー分の領域を持つロケーション・テーブルを作成する。ｈ個のエントリーの、先頭アドレスと最終アドレスが確定する。その双方のアドレスと、ロケーション・テーブルの追加であることを示す識別を、プライマリー・システム０からアクセラレーター・システム１に送信する。このトランザクションに基づき、アクセラレーター・システム１では、ロケーション・テーブルＬ１の追加作成を行う。次に、プライマリー・システム０のロケーション・テーブルＬ０の「ｋ＋１」のエントリーが更新されることになる。何故ならば、ｋ番目のエントリーに格納できなくなり、追加ロケーション・テーブルを作成したからである。

【００６４】

そこで、「ｋ＋１」のＡログをプライマリー・システム０からアクセラレーター・システム１へ送信する。このＡログの送信と、Ａログに基づくアクセラレーター・システム１の更新は、通常の更新と同様である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトラン

ザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

【００６５】

レコードの挿入に伴い、プライマリー・ブロックにオーバーフロー・ブロックを追加する場合は、次の様になる。

【００６６】

ロケーション・テーブルＬ０に、当該ブロックに格納されているレコードの主キー値の最大値、最小の両方または何れか一方を保持しない場合は全く問題ない。ロケーション・テーブルＬ０のエントリーに変更が発生しないため、プライマリー・システム０からアクセラレーター・システム１へ情報を送信する必要はない。

ロケーション・テーブルＬ０に主キー値の最小値、最大値の両方、若しくは、何れか一方を保持している場合で、最小値または最大値が変更されなければ、同様に、プライマリー・システム０からアクセラレーター・システム１へ情報を送信する必要はない。

【００６７】

ロケーション・テーブルＬ０に主キー値の最小値、最大値の両方、若しくは、何れか一方を保持している場合で、最小値または最大値の何れかが変更された場合には、エントリー情報の変更になるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーの更新後の内容を、エントリーの番号と共に、アクセラレーター・システム１に送信する。

【００６８】

[代替キー・テーブル・エントリーの更新をアクセラレーター・システムに反映する方式]

次に、代替キー・テーブルの更新に関して述べる。代替キー・テーブルは、代替キー・ロケーション・テーブルと代替キー・プライマリー・ブロックと代替キー・オーバーフロー・ブロックとで構成されている。代替キー・オーバーフロー・ブロックは、代替キー・プライマリー・ブロックに従属するものとして位置付けられ、代替キー・ロケーション・テーブルで管理されているのは、代替キー・

プライマリー・ブロックのみである。

【００６９】

ここで、代替キー・エントリーの追加があった場合に関して、まず述べる。代替キー・プライマリー・ブロックには図５で示すようなエントリーが格納されている。この代替キー・プライマリー・ブロックに対して、エントリー「ｙ」がエントリー「ｘ」の直後に追加される場合は、代替キー・プライマリー・ブロックの内容は更新されるが、当該代替キー・プライマリー・ブロックの最大値、最小値の変更は無いし、代替キー・オーバーフロー・ブロックの追加も発生しないので、代替キー・ロケーション・テーブルの更新は発生しない。よって、プライマリー・システム０から、アクセラレーター・システム１に対する、更新情報の送信は無い。

【００７０】

次に、代替キー・プライマリー・ブロックのみの状態から、エントリーの挿入により、代替キー・オーバーフロー・ブロックが作成される場合について述べる。図６を用いて説明する。この場合、プライマリー・システムにおいて、目的の代替キー・プライマリー・ブロック「ｐ」に代替キー・エントリー「ｙ」を格納しようとしたが、代替キー・プライマリー・ブロック「ｐ」に十分な空きスペースが無く格納できない状態となる。プライマリー・システムでは、代替キー・オーバーフロー・ブロックの追加を行う必要がある。これは、まず、データ格納エリアで代替キー・オーバーフロー・ブロックを作成するスペースを探し、領域の確保を行う。代替キー・オーバーフロー・ブロックの大きさは、代替キーの種類毎に一定の大きさである。

【００７１】

その後、プライマリー・システム０では、新たな代替キー・エントリー「ｙ」を代替キー・プライマリー・ブロックの代替キー・エントリー「ｚ」があった場所に格納を行い、それに伴いエントリー「ｚ」が代替キー・オーバーフロー・ブロックに格納される。代替キー・プライマリー・ブロックの代替キー値の最小値と最大値の内、最大値が「ｙ」になったため、最大値の情報を「ｙ」に書き換える。また、代替キー・オーバーフロー・ブロックの代替キー値の最小値、最大値

を「ｚ」とする。

【００７２】

この場合も、代替キー・ロケーション・テーブルのエントリーの変更は無いため、プライマリー・システム０から、アクセラレーター・システム１に対する、更新情報の送信は無い。

【００７３】

次に、代替キー・ロケーション・テーブルのエントリーが更新される場合に関して述べる。図７に示すように、更新前の代替キー・プライマリー・ブロックの最後には、「ｘ」と「ｙ」の代替キー・エントリーが格納されている。データの更新・追加によって、新たに代替キー・エントリー「ｚ」が追加されると、この代替キー・プライマリー・ブロックの「ｙ」の後に、代替キー・エントリー「ｚ」が追加されることになる。

【００７４】

この場合、代替キー・プライマリー・ブロック「ｐ」の代替キー値の最大値が、「ｙ」から「ｚ」に更新されたことになる。この場合は、プライマリー・システム０から、アクセラレーター・システム１に対して、Ａログとして、代替キー・エントリーの更新後の値と、どの代替キー・ブロックに対する更新であるかの情報、この場合は代替キー・ブロック番号「ｐ」を送信する。この送信は同期密結合方式を使用して送信される。

【００７５】

アクセラレーター・システム１では、Ａログを受信したら、直ちに代替キー・ロケーション・テーブルの当該エントリーを排他し、当該代替キー・エントリーの更新を行った後、排他を解除し、更新完了をプライマリー・システム０に通知する。

【００７６】

[プライマリー・システムでのアクセス方式]

[主キーによる検索：プライマリー・システムの場合]

プライマリー・システム０における主キーを用いた検索は、「データ格納検索方式で述べてあるとおりである。

検索は、プライマリー・システム０のロケーション・テーブルＬ０を用いて、バイナリー・サーチを行い、ターゲット・キー値が含まれるロケーション・テーブル・エントリーを検索する。

【００７７】

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）にレコードが格納されてい得る。

若しくは、次のロケーション・テーブル・エントリーの最小値より小さくて、当該ロケーション・テーブル・エントリーの最大値より大きい場合には、ターゲット・キー値を持つレコードはデータベース上に存在しないことになる。

【００７８】

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の何れか一方を保持している場合には、前後のロケーション・テーブル・エントリーの主キー値（最小値または最大値）と比較して、対象ブロックを決定する。

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはアクセラレーター・システム１には存在せず、プライマリー・システム０にのみ存在する。

そこで、ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。ブロック「ｎ」にターゲット・キー値を持つレコードが存在すれば、そのレコードがターゲット・レコードとなる。ブロック「ｎ」にターゲット・キー値を持つレコードが存在しなければ、そのレコードが存在しないということになる。

検索の場合は、レコードの更新が発生しないため、ロケーション・テーブルの更新も発生せず、プライマリー・システムからアクセラレーター・システムに対

して情報の送信を行うことは無い。また、原則的に排他は不要である。

【００７９】

[代替キーによる検索：プライマリー・システムの場合]

プライマリー・システム０における代替キーを用いた検索は、「データ格納検索方式で述べてあるとおりである。.

ターゲット・キー値（検索の目的となるキーの値）によって検索を行う。検索は、プライマリー・システム０の代替キー・ロケーション・テーブルＬ０を用いて、バイナリー・サーチを行い、ターゲット・キー値が含まれる代替キー・ロケーション・テーブル・エントリーを検索する。

【００８０】

代替キー・ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの代替キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該代替キー・ロケーション・テーブル・エントリーが管理しているブロック（当該代替キー・プライマリー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合には、代替キー・オーバーフロー・ブロックも含む）に代替キー・エントリーが格納されてい得る。

若しくは、次の代替キー・ロケーション・テーブル・エントリーの最小値より小さくて、当該代替キー・ロケーション・テーブル・エントリーの最大値より大きい場合には、ターゲット・キー値を持つレコードはデータベース上に存在しないことになる。

【００８１】

代替キー・ロケーション・テーブル・エントリーに、その代替キー・エントリーが管理している代替キー・ブロックの代替キー値の最小値と最大値の何れか一方を保持している場合には、前後の代替キー・ロケーション・テーブル・エントリーの代替キー値（最小値または最大値）と比較して、対象代替キー・ブロックを決定する。

【００８２】

また、代替キー・ロケーション・テーブルには、同一の代替キー値を持つ代替キー・ロケーション・テーブル・エントリーが複数存在する可能性がある。これは、代替キーがノン・ユニークでよいという属性を持っているからである。代替キー・ロケーション・テーブル・エントリーが複数存在する場合には、以下の動作を同一の代替キー値を持つ代替キー・ロケーション・テーブル・エントリーの数だけ実行する。

【００８３】

検索した代替キー・エントリーが、代替キー・ブロック番号「ｎ」を指していたとする。そこで、プライマリー・システム０上の代替キー・ブロック「ｎ」にターゲット・キー値を持つ代替キー・エントリーが存在すれば、その代替キー・エントリーが目的のエントリーとなる。ブロック「ｎ」にターゲット・キー値を持つ代替キー・エントリーが存在しなければ、そのターゲット・キー値を持つレコードが存在しないということになる。

これまでで、代替キー・エントリーの検索を終了した。代替キー・エントリーには、代替キー値とその代替キー値を持つレコードの主キー値の他、必要に応じて、レコードが格納されているブロックの番号、またはブロックのアドレスを保持している。

目的のレコードを検索するためには、ロケーション・テーブルＬ０において、ターゲット・レコードを含むブロックを探す必要がある。

【００８４】

代替キー・エントリーにブロック番号を保持している場合は、ロケーション・テーブルＬ０のエントリーの長さと、ブロック番号を掛けたものが、ロケーション・テーブルＬ０の先頭からの変位となる。そうして検索したロケーション・テーブル・エントリーが指しているブロックに目的のレコードが格納されていることになる。当該ブロック内で、検索で求めた代替キー・エントリーが指している主キー値を持つレコードの検索を行う。この場合には、レコードが見つからないという状態は発生しない。

【００８５】

代替キー・エントリーにブロック・アドレスを保持している場合は、代替キー

・エントリーが指しているブロック・アドレスのブロックに目的のレコードが格納されていることになる。当該ブロック内で、検索で求めた代替キー・エントリーが指している主キー値を持つレコードの検索を行う。この場合には、レコードが見つからないという状態は発生しない。

【００８６】

ブロック番号及びブロック・アドレスの何れも保持していない形式である場合には、検索で求めた代替キー・エントリーが挿している主キー値を持つレコードの検索を行うため、まず、アクセラレーター・システム１のロケーション・テーブルＬ１上で、当該主キー値をターゲット・キー値としてバイナリー・サーチを行い、ロケーション・テーブル・エントリーを検索する。そこで求めたブロック番号を元に、プライマリー・システムのブロック内で対象レコードの検索を行う。この場合には、レコードが見つからないという状態は発生しない。

検索の場合は、レコードの更新が発生しないため、ロケーション・テーブルの更新も発生せず、プライマリー・システムからアクセラレーター・システムに対して情報の送信を行うことは無い。また、原則的に排他は不要である。

【００８７】

[レコードの追加、更新、削除：プライマリー・システムの場合]

以上で主キーおよび代替キーでの検索の場合を述べた。検索で用いた方式を応用することで、レコードの追加、更新、削除が行える。

レコードの追加は主キーによって実行される。代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。更新と削除を代替キーで実行する場合には、同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、更新を行うか否か、削除を行うか否かの選択を行うことになる。

【００８８】

[レコードの追加：プライマリー・システムの場合]

まず、レコードの追加の場合に関して記述する。

レコードを追加する場合には、そのレコードがどのブロックに格納されるかを見つける必要がある。「データ格納検索方式」では、レコードはブロック内に主

キー値の順番で格納され、ブロック間では、ブロック番号の大きいブロックに格納されているレコードの主キー値は、小さいブロック番号のブロックに格納されているレコードの主キー値より大きくなっているからである。
追加しようとする場合には、ロケーション・テーブルの検索により、対象ブロックを検索する。

【００８９】

プライマリー・システム０上で行う場合には、まず、ロケーション・テーブルＬ０をバイナリー・サーチして、追加するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する。ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・ギー値が当該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）にレコードを格納することになる。

【００９０】

若しくは、次のロケーション・テーブル・エントリーの最小値より小さくて、当該ロケーション・テーブル・エントリーの最大値より大きい場合にも、ターゲット・キー値を持つレコードは当該ブロックに格納することになる。
ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の何れか一方を保持している場合には、前後のロケーション・テーブル・エントリーの主キー値（最小値または最大値）と比較して、対象ブロックを決定する。

【００９１】

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックプライマリー・システム０に存在しているので、それに対してアクセスする。
ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので

、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。この排他に関しては、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式が高速化のためには好ましい。

【００９２】

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値より小さい主キー値の内、最大の主キー値を持つレコードを見つける。そのレコードの次のアドレスに、当該レコードを挿入することになる。レコードを挿入するためには、挿入するレコードより、上位アドレスにあるレコードを挿入レコードが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、オーバーフロー・ブロックが発生する場合には、オーバーフロー・ブロックの追加を行い、レコードの移動を行う。その後、レコードの追加を行う。

【００９３】

このようにして、レコードの追加が行われるが、レコードが挿入となる場合には、ロケーション・テーブルＬ０の変更は行われないので、ロケーション・テーブルＬ０の変更をプライマリー・システム０からアクセラレーター・システム１に通知する必要は無い。

このレコードの追加によって、ロケーション・テーブルが変更される場合は、追加されるレコードの主キー値が当該ブロック中で最大のキー値を持つ場合である。この場合には、ロケーション・テーブルの当該ロケーション・テーブル・エントリーの情報の更新を行う。それに伴って、プライマリー・システムからアクセラレーター・システム１に対して当該ロケーション・テーブル・エントリーの更新情報を送信する。更新後のロケーション・テーブル・エントリーの内容（Ａログ）と、どのロケーション・テーブル・エントリーであるかの識別情報（ブロック番号）が情報である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新

が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

【００９４】

レコード追加の特殊な場合として、最終ブロックの次に新たにブロックを作成して、そこにレコードを格納する場合があるが、新たにブロックを追加し、そのブロックを管理するためにロケーション・テーブルに新たにエントリーを追加する必要があるが、それ以降は、上述の追加の場合と同様である。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

【００９５】

[レコードの追加に伴う代替キー・エントリーの追加：プライマリー・システムの場合]

ところが、レコードが追加されると、それに伴って代替キー・エントリーの追加が行われる。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。

代替キーが３種類ある場合には、レコード追加によって３種類の代替キー・エントリーの追加が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは省略する。

【００９６】

代替キー・エントリーがＡｅ、Ｂｅ、Ｃｅの３つ作成される。この各々を代替キー・テーブルＡ，Ｂ，Ｃに格納する必要がある。代替キー・テーブルＡ０の場合に関して述べる。代替キー・ロケーション・テーブルＡＬ０をＡｅの代替キー値をターゲット・キー値としてバイナリー・サーチする。

代替キー・ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当

該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）に当該代替キー・エントリーを格納することになる。

若しくは、次の代替キー・ロケーション・テーブル・エントリーの最小値より小さくて、当該代替キー・ロケーション・テーブル・エントリーの最大値より大きい場合にも、ターゲット・キー値を持つ代替キー・エントリーは当該ブロックに格納することになる。

【００９７】

代替キー・ロケーション・テーブル・エントリーに、そのエントリーが管理している代替キー・ブロックの代替キー値の最小値と最大値の何れか一方を保持している場合には、前後の代替キー・ロケーション・テーブル・エントリーの代替キー値（最小値または最大値）と比較して、対象ブロックを決定する。

目的の代替キー・ロケーション・テーブル・エントリーの検索を完了したら、当該代替キー・ロケーション・テーブル・エントリーと、当該代替キー・ロケーション・テーブル・エントリーが指している代替キー・ブロックの排他を実行する。

この排他順序は、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で述べてあるように、ロケーション・テーブル、ブロック、代替キー・ロケーション・テーブル、代替キー・ブロックの排他順序が一定であるので、デッドロックが発生する可能性は、非常に少なくなる。

【００９８】

ここで、Ａｅの挿入を行うのであるが、対象代替キー・ブロックの中の代替キー・エントリーを調べる。代替キーはノン・ユニークであるので、同一の代替キー値を持つエントリーが存在する場合がある。

まず、ターゲット・キー値と同一の代替キー値を持つ代替キー・エントリーが存在する場合に関して述べる。この場合は、同一の代替キー値を持つ代替キー・エ

ントリーの内、追加対象代替キー・エントリーの主キーに着目し、追加対象代替キー・エントリーの主キー値より小さい主キー値の内、最大の主キー値を持つ代替キー・エントリーの次のアドレスに、当該代替キー・エントリーを挿入することになる。代替キー・エントリーを挿入するためには、挿入する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを追加対象代替キー・エントリーが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、代替キー・オーバーフロー・ブロックが発生する場合には、代替キー・オーバーフロー・ブロックの追加を行い、代替キー・エントリーの移動を行う。その後、代替キー・エントリーの追加を行う。

【００９９】

次に、ターゲット・キー値と同一の代替キー値を持つ代替キー・エントリーが存在しない場合に関して述べる。この場合は、対象ブロック内の代替キー値を持つ代替キー・エントリーの内、追加対象代替キー・エントリーの代替キー値より小さい主キー値の内、最大の代替キー値を持つ代替キー・エントリーの次のアドレスに、当該代替キー・エントリーを挿入することになる。代替キー・エントリーを挿入するためには、挿入する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを追加対象代替キー・エントリーが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、代替キー・オーバーフロー・ブロックが発生する場合には、代替キー・オーバーフロー・ブロックの追加を行い、代替キー・エントリーの移動を行う。その後、代替キー・エントリーの追加を行う。

【０１００】

この代替キー・エントリーの追加によって、代替キー・ロケーション・テーブルが変更される場合は、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。更新後の代替キー・ロケーション・テーブル・エントリーの内容（Ａログ

）と、どの代替キー・ロケーション・テーブル・エントリーであるかの識別情報（ブロック番号）が情報である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同様に実施する。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加が完了したら、これまでの排他を解除する。

【０１０１】

[主キーによるレコードの更新：プライマリー・システムの場合]

プライマリー・システム０上で行う場合には、まず、ロケーション・テーブルＬ０をバイナリー・サーチして、更新するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する。ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）にレコードを格納されてい得ることになる。

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の何れか一方を保持している場合には、前後のロケーション・テーブル・エントリーの主キー値（最小値または最大値）と比較して、対象ブロックを決定する。

【０１０２】

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式を使用することが、排他の高速化のために好ましい。

【０１０３】

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。そのレコードを更新処理することになる。また、ブロック内にターゲット・キー値を持つレコードが存在しない場合には、レコードが存在しないということになる。

このようにして、レコードの更新が行われるが、主キー値の更新は許されないので、ロケーション・テーブルＬ０の変更は行われない。主キー値の更新を行いたい場合は、当該主キー値を持ったレコードを一旦削除し、新たな主キー値を持ったレコードを追加すると言う方法を採る。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

【０１０４】

[主キーによるレコードの更新に伴う代替キー・エントリーの追加、削除：プライマリー・システムの場合]

ところが、レコードが更新されると、それに伴って代替キー・エントリーの更新、実際には追加と削除が行われる場合がある。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード更新の場合は、必ず代替キー・エントリーの更新が行われるわけではなく、レコードの項目（フィールド）で代替キーに指定した項目が更新された場合に、代替キー・エントリーの更新が必要となる。

【０１０５】

代替キーが３種類ある場合には、レコード更新によって０種類から３種類の代替キー・エントリーの更新が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは省略する。

ここでは、すべての代替キー・エントリーが更新される場合を例にとって説明する。新しい代替キー・エントリーがＡｅｎ、Ｂｅｎ、Ｃｅｎの３つ作成され、これらの代替キー・エントリーの追加が必要になる。逆に、当該レコードのそれまでの代替キー・エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏは不要となり、削除が必要になる。

【０１０６】

Ａｅｎ、Ｂｅｎ、Ｃｅｎの３つの代替キー・エントリーの各々を代替キー・テーブルＡ，Ｂ，Ｃに格納する必要がある。代替キー・テーブルＡ０の場合に関して述べる。代替キー・ロケーション・テーブルＡＬ０をＡｅｎの代替キー値をターゲット・キー値としてバイナリー・サーチする。

これ以下の動作は、レコード追加による代替キー・エントリーの追加の場合と同様であるので、省略する。また、Ｂｅｎ、Ｃｅｎに関しては、Ａｅｎの場合と同様であるので、これも省略する。

【０１０７】

次に、代替キー・エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏの各々を、代替キー・テーブルＡ，Ｂ，Ｃから削除する必要がある。前記の説明と同様に、Ａｅｏの場合に関して説明を行う。やはり、代替キー・ロケーション・テーブルＡＬ０を使用して、Ａｅｏの中にある代替キー値をターゲット・キー値として、バイナリー・サーチを行う。

代替キー・ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの代替キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該代替キー・ロケーション・テーブル・エントリーが管理してい

る代替キー・ブロック（当該代替キー・プライマリー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合には、代替キー・オーバーフロー・ブロックも含む）に当該代替キー・エントリーが格納されていることになる。

【０１０８】

代替キー・ロケーション・テーブル・エントリーに、そのエントリーが管理している代替キー・ブロックの代替キー値の最小値と最大値の何れか一方を保持している場合には、前後の代替キー・ロケーション・テーブル・エントリーの代替キー値（最小値または最大値）と比較して、対象ブロックを決定する。

目的の代替キー・ロケーション・テーブル・エントリーの検索を完了したら、当該代替キー・ロケーション・テーブル・エントリーと、当該代替キー・ロケーション・テーブル・エントリーが指している代替キー・ブロックの排他を実行する。

この排他順序は、「データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で述べてあるように、ロケーション・テーブル、ブロック、代替キー・ロケーション・テーブル、代替キー・ブロックの排他順序が一定であるので、デッドロックが発生する可能性は、非常に少なくなる。

ここで、Ａｅｏの削除を行うのであるが、対象代替キー・ブロックの中の代替キー・エントリーを調べる。代替キーはノン・ユニークであるので、同一の代替キー値を持つ代替キー・エントリーが存在する場合があるので、代替キー値と主キー値が一致する代替キー・エントリーを見つける。

代替キー・エントリーが見つからないということは有り得ない。

【０１０９】

次に、ターゲット・キー値と同一の代替キー値と主キー値を持つ代替キー・エントリーの削除を行う。代替キー・エントリーを削除すると、削除した代替キー・エントリーが占めていた領域が空くことになる。このままでは、その後の代替キーによる検索に支障を来たすので、削除する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを削除対象代替キー・エントリーが占めていた領域の分だけアドレスの下位方向に動かしておく必要がある。また、下位方向に移動することによって、代替キー・オーバーフロー・ブロックにあった代替

キー・エントリーが、それよりも前の代替キー・オーバーフロー・ブロックまたは代替キー・プライマリー・ブロックに格納されてしまう場合には、当該代替キー・オーバーフロー・ブロックが不要となるが、この場合は、当該代替キー・オーバーフロー・ブロックを未使用ブロックとしてもよいし、そのまま接続しておいてもよい。接続しておく場合の、当該代替キー・オーバーフロー・ブロックの代替キー値の最小値と最大値は、直前の代替キー・プライマリー・ブロックまたは代替キー・オーバーフロー・ブロックの代替キー値と、次の、代替キー・ロケーション・テーブル・エントリーの代替キーの最小値と矛盾しない範囲で、最小値と最大値を同一にしておくことが好ましい。

【０１１０】

この代替キー・エントリーの追加と削除によって、代替キー・ロケーション・テーブルが変更される場合は、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合と、削除される代替キー値が当該ブロック中で最小または最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。更新後の代替キー・ロケーション・テーブル・エントリーの内容（Ａログ）と、どの代替キー・ロケーション・テーブル・エントリーであるかの識別情報（ブロック番号）が情報である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同様に実施する。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加・削除が完了したら、これまでの排他を解除する。

【０１１１】

［代替キーによるレコードの更新：プライマリー・システムの場合］

プライマリー・システム０上で行う場合には、まず、代替キー・ロケーション・テーブルＡＬ０をバイナリー・サーチして、更新するレコードの代替キー値を持つ代替キー・ロケーション・テーブル・エントリーを見つけ出す。

代替キー・ロケーション・テーブル・エントリーを検索した後、代替キー・ブロック内を検索し、代替キー・エントリーを検索する。代替キー・エントリーが検索できたら、ロケーション・テーブルＬ０を使用して、バイナリー・サーチを行い、レコードが格納されているブロックを検索する。

この動作は、代替キーによる検索で述べた方法と同様である。

代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。更新を代替キーで実行する場合には、同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、更新を行うか否かの選択を行うことになる。ここでは、複数のレコードを順次更新するものとする。

【０１１２】

検索したロケーション・テーブル・エントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にあるブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式を使用することが、排他の高速化のために好ましい。

【０１１３】

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。ブロック内にターゲット・キー値を持つレコードが存在

しない場合は有り得ない。

このようにして、レコードの更新が行われるが、主キー値の更新は許されないので、ロケーション・テーブルＬ０の変更は行われない。主キー値の更新を行いたい場合は、当該主キー値を持ったレコードを一旦削除し、新たな主キー値を持ったレコードを追加すると言う方法を採る。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

【０１１４】

[代替キーによるレコードの更新に伴う代替キー・エントリーの追加、削除：プライマリー・システムの場合]

レコードが更新されると、それに伴って代替キー・エントリーの更新、実際には追加と削除が行われる場合がある。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード更新の場合は、必ず代替キー・エントリーの更新が行われるわけではなく、レコードの項目（フィールド）で代替キーに指定した項目が更新された場合に、代替キー・エントリーの更新が必要となる。代替キーが３種類ある場合には、レコード更新によって０種類から３種類の代替キー・エントリーの更新が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは省略する。

ここでは、すべての代替キー・エントリーが更新される場合を例にとって説明する。新しい代替キー・エントリーがＡｅｎ、Ｂｅｎ、Ｃｅｎの３つ作成され、これらの代替キー・エントリーの追加が必要になる。逆に、当該レコードのそれまでの代替キー・エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏは不要となり、削除が必要になる。

この代替キー・エントリーの追加・削除は、「主キーによるレコード更新による、代替キー・エントリーの更新」と方式が同様であるので、説明は省略する。

【０１１５】

[主キーによるレコードの削除：プライマリー・システムの場合]

プライマリー・システム０上で行う場合には、まず、ロケーション・テーブルＬ０をバイナリー・サーチして、削除するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する。ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）にレコードを格納されてい得ることになる。

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の何れか一方を保持している場合には、前後のロケーション・テーブル・エントリーの主キー値（最小値または最大値）と比較して、対象ブロックを決定する。

【０１１６】

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式を使用することが、排他の高速化のために好ましい。

【０１１７】

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。そのレコードを削除処理することになる。ブロック内にターゲット・キー値を持つレコードが存在しない場合は、当該レコードは存在し

ないことになる。

このようにして、レコードの削除が行われる。レコードを削除すると、削除したレコードが占めていた領域が空くことになる。このままでは、その後の主キーによる検索に支障を来たすので、削除するレコードより上位アドレスにあるレコードを削除対象レコードが占めていた領域の分だけアドレスの下位方向に動かしておく必要がある。また、下位方向に移動することによって、オーバーフロー・ブロックにあったレコードが、それよりも前のオーバーフロー・ブロックまたはプライマリー・ブロックに格納されてしまう場合には、オーバーフロー・ブロックが不要となるが、この場合は、当該オーバーフロー・ブロックを未使用ブロックとしてもよいし、そのまま接続しておいてもよい。接続しておく場合の、当該ブロックの主キー値の最小値と最大値は、直前のプライマリー・ブロックまたはオーバーフロー・ブロックの主キー値と、次の、ロケーション・テーブル・エントリーの主キーの最小値と矛盾しない範囲で、最小値と最大値を同一にしておくことが好ましい。

【０１１８】

このレコードの削除によって、ロケーション・テーブルが変更される場合は、削除される主キー値が当該ブロック中で最小または最大のキー値を持つ場合で且つ、代替キー・エントリーに主キー値の最小値または最大値を持っている場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

【０１１９】

[主キーによるレコードの削除に伴う代替キー・エントリーの削除：プライマリー・システムの場合]

ところが、レコードが削除されると、それに伴って代替キー・エントリーの更新、実際には削除が行われる。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード削除の場合は、必ず代替キー・エントリーの更新

が行われる。

代替キーが３種類ある場合には、レコード削除によって３種類の代替キー・エントリーの削除が行われる。

これまでの代替キー・エントリーＡｅ、Ｂｅ、Ｃｅの３つが削除対象となる。

【０１２０】

代替キー・エントリーの削除は、代替キー・エントリーの追加と削除の中の削除に関するところで述べた説明と同様であるので、ここでは省略する。

この代替キー・エントリーの追加と削除によって、代替キー・ロケーション・テーブルが変更される場合は、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合と、削除される代替キー値が当該ブロック中で最小または最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。更新後の代替キー・ロケーション・テーブル・エントリーの内容（Ａログ）と、どの代替キー・ロケーション・テーブル・エントリーであるかの識別情報（ブロック番号）が情報である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同様に実施する。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加・削除が完了したら、これまでの排他を解除する。

【０１２１】

[代替キーによるレコードの削除：プライマリー・システムの場合]

プライマリー・システム０上で行う場合には、まず、代替キー・ロケーション・テーブルＡＬ０をバイナリー・サーチして、削除するレコードの代替キー値を

持つ代替キー・ロケーション・テーブル・エントリーを検索する。

代替キー・ロケーション・テーブル・エントリーを検索した後、プライマリー・システム上の代替キー・ブロック内を検索し、代替キー・エントリーを見つけ出す。代替キー・エントリーが検索できたら、ロケーション・テーブルＬ０を使用して、バイナリー・サーチを行い、レコードが格納されているブロックを検索する。

この動作は、代替キーによるレコードの検索で述べた方法と同様である。

代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。削除を代替キーで実行する場合には、同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、削除を行うか否かの選択を行うことになる。ここでは、複数のレコードを順次削除するものとする。

【０１２２】

検索したロケーション・テーブル・エントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にあるブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「データ及びデータベースの無停止自動再編成システム並びにデータ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式を使用することが、排他の高速化のために好ましい。

【０１２３】

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。ブロック内にターゲット・キー値を持つレコードが存在しない場合は有り得ない。

この後、「主キーによるレコードの削除」の場合と同様に、ブロック内で削除

レコードが占めていた領域を詰め、必要に応じてロケーション・テーブル・エントリーの更新を行い、代替キー・エントリーの削除を行う。

このようにして、レコードの削除が行わる。

一連の動作が完了したら、排他解除を行う。

以上で、プライマリー・システム内でのデータ（レコード）の検索、追加、更新、削除に伴うプライマリー・システム内での動作と、それに伴う、アクセラレーター・システムの動作に関して述べた。

【０１２４】

[アクセラレーター・システムを利用したアクセスに関して]

次に、アクセラレーター・システム内で、データの検索、追加、更新、削除に伴うアクセラレーター・システム内での動作と、それに伴うプライマリー・システム内での動作、更にプライマリー・システムでの動作に伴う、アクセラレーター・システムの動作に関して説明を行う。

【０１２５】

[主キーによる検索：アクセラレーター・システム１の場合]

アクセラレーター・システム１における主キーを用いた検索の場合を説明する。ターゲット・キー値（検索の目的となるキーの値）によって検索を行う。検索は、アクセラレーター・システム１のロケーション・テーブルＬ１を用いて、バイナリー・サーチを行い、ターゲット・キー値が含まれるロケーション・テーブル・エントリーを検索する。

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該代替キー・　ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが指しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む。ブロックはプライマリー・システム上に存在する）にレコードが格納されてい得る。

若しくは、次のロケーション・テーブル・エントリーの最小値より小さくて、

当該ロケーション・テーブル・エントリーの最大値より大きい場合には、ターゲット・キー値を持つレコードはデータベース上に存在しないことになる。

【０１２６】

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の何れか一方を保持している場合には、前後のロケーション・テーブル・エントリーの主キー値（最小値または最大値）と比較して、対象ブロックを決定する。

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはアクセラレーター・システム１には存在せず、プライマリー・システム０にのみ存在する。

そこで、ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。ブロック「ｎ」にターゲット・キー値を持つレコードが存在すれば、そのレコードがターゲット・レコードとなる。ブロック「ｎ」にターゲット・キー値を持つレコードが存在しなければ、そのレコードが存在しないということになる。

検索の場合は、レコードの更新が発生しないため、ロケーション・テーブルの更新も発生せず、プライマリー・システムからアクセラレーター・システムに対して情報の送信を行うことは無い。また、原則的に排他は不要である。

【０１２７】

[代替キーによる検索：アクセラレーター・システム１の場合]

アクセラレーター・システム１における代替キーを用いた検索の場合を説明する。ターゲット・キー値（検索の目的となるキーの値）によって検索を行う。検索は、アクセラレーター・システム１の代替キー・ロケーション・テーブルＡＬ１を用いて、バイナリー・サーチを行い、ターゲット・キー値が含まれる代替キー・ロケーション・テーブル・エントリーを検索する。

代替キー・ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの代替キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間であ

る場合には、当該代替キー・ロケーション・テーブル・エントリーが管理しているブロック（当該代替キー・プライマリー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合には、代替キー・オーバーフロー・ブロックも含む）に代替キー・エントリーが格納されてい得る。

若しくは、次の代替キー・ロケーション・テーブル・エントリーの最小値より小さくて、当該代替キー・ロケーション・テーブル・エントリーの最大値より大きい場合には、ターゲット・キー値を持つレコードはデータベース上に存在しないことになる。

【０１２８】

代替キー・ロケーション・テーブル・エントリーに、その代替キー・エントリーが管理している代替キー・ブロックの代替キー値の最小値と最大値の何れか一方を保持している場合には、前後の代替キー・ロケーション・テーブル・エントリーの代替キー値（最小値または最大値）と比較して、対象代替キー・ブロックを決定する。

また、代替キー・ロケーション・テーブルには、同一の代替キー値を持つ代替キー・ロケーション・テーブル・エントリーが複数存在する可能性がある。これは、代替キーがノン・ユニークでよいという属性を持っているからである。代替キー・ロケーション・テーブル・エントリーが複数存在する場合には、以下の動作を同一の代替キー値を持つ代替キー・ロケーション・テーブル・エントリーの数だけ実行する。

【０１２９】

検索した代替キー・エントリーが、代替キー・ブロック番号「ｎ」を指していたとする。代替キー・ブロックはアクセラレーター・システム１には存在せず、プライマリー・システム０にのみ存在する。

そこで、代替キー・ブロックの代替キー・エントリーに対するアクセスは、プライマリー・システム０にある代替キー・ブロック「ｎ」に対して行うことになる。代替キー・ブロック「ｎ」にターゲット・キー値を持つ代替キー・エントリーが存在すれば、その代替キー・エントリーが目的のエントリーとなる。ブロック「ｎ」にターゲット・キー値を持つ代替キー・エントリーが存在しなければ、

そのターゲット・キー値を持つレコードが存在しないということになる。

これまでで、代替キー・エントリーの検索を終了した。代替キー・エントリーには、代替キー値とその代替キー値を持つレコードの主キー値の他、必要に応じて、レコードが格納されているブロックの番号、またはブロックのアドレスを保持している。

目的のレコードを検索するためには、再度、アクセラレーター・システム１で、ロケーション・テーブルＬ１を使用して、ターゲット・レコードを含むブロックを探す必要がある。

【０１３０】

代替キー・エントリーにブロック番号を保持している場合は、アクセラレーター・システム１のロケーション・テーブルＬ１を直接的に検索する。ロケーション・テーブルＬ１のエントリーの長さと、ブロック番号を掛けたものが、ロケーション・テーブルＬ１の先頭からの変位となるからである。そうして検索したロケーション・テーブル・エントリーが指しているブロックに目的のレコードが格納されていることになる。ブロックはプライマリー・システム０のブロックを参照する。当該ブロック内で、検索で求めた代替キー・エントリーが指している主キー値を持つレコードの検索を行う。この場合には、レコードが見つからないという状態は発生しない。

代替キー・エントリーにブロック・アドレスを保持している場合は、アクセラレーター・システム１のロケーション・テーブルＬ１を見る必要はなく代替キー・エントリーが指しているブロック・アドレスのブロックに目的のレコードが格納されていることになる。ブロックはプライマリー・システム０のブロックを参照する。当該ブロック内で、検索で求めた代替キー・エントリーが指している主キー値を持つレコードの検索を行う。この場合には、レコードが見つからないという状態は発生しない。

【０１３１】

ブロック番号及びブロック・アドレスの何れも保持していない形式である場合には、検索で求めた代替キー・エントリーが挿している主キー値を持つレコードの検索を行うため、まず、アクセラレーター・システム１のロケーション・テー

ブルＬ１上で、当該主キー値をターゲット・キー値としてバイナリー・サーチを行い、ロケーション・テーブル・エントリーを検索する。そこで求めたブロック番号を元に、プライマリー・システム０のブロック内で対象レコードの検索を行う。この場合には、レコードが見つからないという状態は発生しない。

検索の場合は、レコードの更新が発生しないため、ロケーション・テーブルの更新も発生せず、プライマリー・システムからアクセラレーター・システムに対して情報の送信を行うことは無い。また、原則的に排他は不要である。

【０１３２】

[レコードの追加、更新、削除：アクセラレーター・システムの場合]

以上で主キーおよび代替キーでの検索の場合を述べた。検索で用いた方式を応用することで、レコードの追加、更新、削除が行える。

レコードの追加は主キーによって実行される。代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。更新と削除を代替キーで実行する場合には、同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、更新を行うか否か、削除を行うか否かの選択を行うことになる。

【０１３３】

まず、レコードの追加の場合に関して記述する。

レコードを追加する場合には、そのレコードがどのブロックに格納されるかを見つける必要がある。「データ格納検索方式」では、レコードはブロック内に主キー値の順番で格納され、ブロック間では、ブロック番号の大きいブロックに格納されているレコードの主キー値は、小さいブロック番号のブロックに格納されているレコードの主キー値より大きくなっているからである。

追加しようとする場合には、アクセラレーター・システム１のロケーション・テーブルＬ１の検索により、対象ブロックを検索する。

【０１３４】

[レコードの追加：アクセラレーター・システムの場合]

アクセラレーター・システム１上でレコードの追加を行う場合には、まず、ロケーション・テーブルＬ１をバイナリー・サーチして、追加するレコードの主キ

一値を持つロケーション・テーブル・エントリーを検索する。ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）にレコードを格納することになる。

若しくは、次のロケーション・テーブル・エントリーの最小値より小さくて、当該ロケーション・テーブル・エントリーの最大値より大きい場合にも、ターゲット・キー値を持つレコードは当該ブロックに格納することになる。

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の何れか一方を保持している場合には、前後のロケーション・テーブル・エントリーの主キー値（最小値または最大値）と比較して、対象ブロックを決定する。

【０１３５】

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはアクセラレーター・システムには存在せず、ブロックプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式が高速化のためには好ましい。

【０１３６】

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値より小さい主キー値

の内、最大の主キー値を持つレコードを見つける。そのレコードの次のアドレスに、当該レコードを挿入することになる。レコードを挿入するためには、挿入するレコードより、上位アドレスにあるレコードを挿入レコードが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、オーバーフロー・ブロックが発生する場合には、オーバーフロー・ブロックの追加を行い、レコードの移動を行う。その後、レコードの追加を行う。

【０１３７】

このようにして、レコードの追加が行われるが、レコードが挿入となる場合には、ロケーション・テーブルＬ０の変更は行われないので、ロケーション・テーブルＬ０の変更をプライマリー・システム０からアクセラレーター・システム１に通知する必要は無い。

【０１３８】

このレコードの追加によって、ロケーション・テーブルが変更される場合は、追加されるレコードの主キー値が当該ブロック中で最大のキー値を持つ場合である。この場合には、プライマリー・システムのロケーション・テーブルＬ０の当該ロケーション・テーブル・エントリーの情報の更新を行う。それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該ロケーション・テーブル・エントリーの更新情報を送信する。更新後のロケーション・テーブル・エントリーの内容（Ａログ）と、どのロケーション・テーブル・エントリーであるかの識別情報（ブロック番号）が情報である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

レコード追加の特殊な場合として、最終ブロックの次に新たにブロックを作成して、そこにレコードを格納する場合があるが、新たにブロックを追加し、そのブロックを管理するためにロケーション・テーブルに新たにエントリーを追加する必要があるが、それ以降は、上述の追加の場合と同様である。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

【０１３９】

[レコードの追加に伴う代替キー・エントリーの追加：アクセラレーター・システムの場合]

ところが、レコードが追加されると、それに伴って代替キー・エントリーの追加が行われる。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。

代替キーが３種類ある場合には、レコード追加によって３種類の代替キー・エントリーの追加が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは非登録の場合の説明を省略する。

代替キー・エントリーがＡｅ、Ｂｅ、Ｃｅの３つ作成される。この各々を代替キー・テーブルＡ，Ｂ，Ｃに格納する必要がある。代替キー・テーブルＡの場合に関して述べる。アクセラレーター・システム１上で、代替キー・ロケーション・テーブルＡＬ１をＡｅの代替キー値をターゲット・キー値としてバイナリー・サーチする。

【０１４０】

代替キー・ロケーション・テーブル・エントリーに、その代替キー・エントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）に当該代替キー・エントリーを格納することになる。

若しくは、次の代替キー・ロケーション・テーブル・エントリーの最小値より小さくて、当該代替キー・ロケーション・テーブル・エントリーの最大値より大

きい場合にも、ターゲット・キー値を持つ代替キー・エントリーは当該代替キー・ブロックに格納することになる。

【０１４１】

代替キー・ロケーション・テーブル・エントリーに、その代替キー・エントリーが管理している代替キー・ブロックの代替キー値の最小値と最大値の何れか一方を保持している場合には、前後の代替キー・ロケーション・テーブル・エントリーの代替キー値（最小値または最大値）と比較して、対象ブロックを決定する。

目的の代替キー・ロケーション・テーブル・エントリーの検索をアクセラレーター・システムで完了したら、その代替キー・ロケーション・テーブル・エントリーに対応する、プライマリー・システム０の当該代替キー・ロケーション・テーブル・エントリーと、当該代替キー・ロケーション・テーブル・エントリーが指している代替キー・ブロックの排他を実行する。つまり、代替キー・ブロックはプライマリー・システム上にのみ存在しているので、代替キー・ブロックに対する操作はプライマリー・システム上で実行する。

この排他順序は、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で述べてあるように、ロケーション・テーブル、ブロック、代替キー・ロケーション・テーブル、代替キー・ブロックの排他順序が一定であるので、デッドロックが発生する可能性は、非常に少なくなる。従来の方式では、この排他順序が逆になる場合があったため、デッドロックの可能性が大きくなっていた。

【０１４２】

これ以下の動作は、プライマリー・システム上での実行となる。ここで、Ａｅの挿入を行うのであるが、対象代替キー・ブロックの中の代替キー・エントリーを調べる。代替キーはノン・ユニークであるので、同一の代替キー値を持つエントリーが存在する場合がある。

まず、ターゲット・キー値と同一の代替キー値を持つ代替キー・エントリーが存在する場合に関して述べる。この場合は、同一の代替キー値を持つ代替キー・エントリーの内、追加対象代替キー・エントリーの主キーに着目し、追加対象代

替キー・エントリーの主キー値より小さい主キー値の内、最大の主キー値を持つ代替キー・エントリーの次のアドレスに、当該代替キー・エントリーを挿入することになる。代替キー・エントリーを挿入するためには、挿入する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを追加対象代替キー・エントリーが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、代替キー・オーバーフロー・ブロックが発生する場合には、代替キー・オーバーフロー・ブロックの追加を行い、代替キー・エントリーの移動を行う。その後、代替キー・エントリーの追加を行う。

【０１４３】

次に、ターゲット・キー値と同一の代替キー値を持つ代替キー・エントリーが存在しない場合に関して述べる。この場合は、対象ブロック内の代替キー値を持つ代替キー・エントリーの内、追加対象代替キー・エントリーの代替キー値より小さい主キー値の内、最大の代替キー値を持つ代替キー・エントリーの次のアドレスに、当該代替キー・エントリーを挿入することになる。代替キー・エントリーを挿入するためには、挿入する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを追加対象代替キー・エントリーが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、代替キー・オーバーフロー・ブロックが発生する場合には、代替キー・オーバーフロー・ブロックの追加を行い、代替キー・エントリーの移動を行う。その後、代替キー・エントリーの追加を行う。

【０１４４】

この代替キー・エントリーの追加によって、代替キー・ロケーション・テーブルが変更される場合は、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルＬ０の当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。
それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。更新後の代替キー・ロケーション・テーブル・エントリーの内容（Ａログ）

と、どの代替キー・ロケーション・テーブル・エントリーであるかの識別情報（ブロック番号）が情報である。

【０１４５】

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同様に実施する。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加が完了したら、これまでの排他を解除する。

【０１４６】

[主キーによるレコードの更新：アクセラレーター・システムの場合]

[主キーによるレコードの更新に伴う、ロケーション・テーブルとブロックの更新：アクセラレーター・システムの場合]

アクセラレーター・システム１上でレコード更新要求に基づく処理を行う場合には、まず、ロケーション・テーブルＬ１をバイナリー・サーチして、更新するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する。ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）にレコードを格納されてい得ることになる。

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の何れか一方を保持している場合には、前後のロケーション・テーブル・エントリーの主キー値（最小値または最大値）と比較して、対象ブロックを決定する。

【０１４７】

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式を使用することが、排他の高速化のために好ましい。

【０１４８】

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つけ、そのレコードを更新処理することになる。また、ブロック内にターゲット・キー値を持つレコードが存在しない場合には、レコードが存在しないということになる。

このようにして、レコードの更新が行われるが、主キー値の更新は許されないので、ロケーション・テーブルＬ０の変更は行われない。主キー値の更新を行いたい場合は、当該主キー値を持ったレコードを一旦削除し、新たな主キー値を持ったレコードを追加すると言う方法を採る。よって、主キー値が「変更」される場合は、レコード追加と削除の説明で十分である。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

【０１４９】

[主キーによるレコードの更新に伴う代替キー・エントリーの追加、削除：アクセラレーター・システムの場合]

レコードが更新されると、それに伴って代替キー・エントリーの更新、実際には追加と削除が行われる場合がある。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存

在する場合に関して述べる。レコード更新の場合は、必ず代替キー・エントリーの更新が行われるわけではなく、レコードの項目（フィールド）で代替キーに指定した項目が更新された場合に、代替キー・エントリーの更新が必要となる。

代替キーが３種類ある場合には、レコード更新によって０種類から３種類の代替キー・エントリーの更新が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは省略する。

ここでは、すべての代替キー・エントリーが更新される場合を例にとって説明する。新しい代替キー・エントリーがＡｅｎ、Ｂｅｎ、Ｃｅｎの３つ作成され、これらの代替キー・エントリーの追加が必要になる。逆に、当該レコードのそれまでの代替キー・エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏは不要となり、削除が必要になる。

【０１５０】

Ａｅｎ、Ｂｅｎ、Ｃｅｎの３つの代替キー・エントリーの各々を代替キー・テーブルＡ，Ｂ，Ｃに格納する必要がある。代替キー・テーブルＡ０の場合に関して述べる。アクセラレーター・システム上で、代替キー・ロケーション・テーブルＡＬ１をＡｅｎの代替キー値をターゲット・キー値としてバイナリー・サーチする。

これ以下の動作は、レコード追加による代替キー・エントリーの追加の場合と同様であるので、省略する。また、Ｂｅｎ、Ｃｅｎに関しては、Ａｅｎの場合と同様であるので、これも省略する。

【０１５１】

次に、代替キー・エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏの各々を、代替キー・テーブルＡ，Ｂ，Ｃから削除する必要がある。前記の説明と同様に、Ａｅｏの場合に関して説明を行う。やはり、アクセラレーター・システム１上で、代替キー・ロケーション・テーブルＡＬ１を使用して、Ａｅｏの中にある代替キー値をターゲット・キー値として、バイナリー・サーチを行う。

代替キー・ロケーション・テーブル・エントリーに、そのエントリーが管理している代替キー・ブロックの代替キー値の最小値と最大値の両方を保持している

場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該代替キー・ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該代替キー・ロケーション・テーブル・エントリーが管理している代替キー・ブロック（当該代替キー・プライマリー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合には、代替キー・オーバーフロー・ブロックも含む）に当該代替キー・エントリーが格納されていることになる。

【０１５２】

代替キー・ロケーション・テーブル・エントリーに、そのエントリーが管理している代替キー・ブロックの代替キー値の最小値と最大値の何れか一方を保持している場合には、前後の代替キー・ロケーション・テーブル・エントリーの代替キー値（最小値または最大値）と比較して、対象ブロックを決定する。

目的の代替キー・ロケーション・テーブル・エントリーの検索を完了したら、プライマリー・システム０の当該代替キー・ロケーション・テーブル・エントリーと、当該代替キー・ロケーション・テーブル・エントリーが指している代替キー・ブロックの排他を実行する。

この排他順序は、「データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で述べてあるように、ロケーション・テーブル、ブロック、代替キー・ロケーション・テーブル、代替キー・ブロックの排他順序が一定であるので、デッドロックが発生する可能性は、非常に少なくなる。

【０１５３】

ここで、Ａｅｏの削除を行うのであるが、対象代替キー・ブロックの中の代替キー・エントリーを調べる。代替キーはノン・ユニークであるので、同一の代替キー値を持つ代替キー・エントリーが存在する場合があるので、代替キー値と主キー値が一致する代替キー・エントリーを見つける。

代替キー・エントリーが見つからないということは有り得ない。

次に、ターゲット・キー値と同一の代替キー値と主キー値を持つ代替キー・エントリーの削除を行う。代替キー・エントリーを削除すると、削除した代替キー・エントリーが占めていた領域が空くことになる。このままでは、その後の代替

キーによる検索に支障を来たすので、削除する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを削除対象代替キー・エントリーが占めていた領域の分だけアドレスの下位方向に動かしておく必要がある。また、下位方向に移動することによって、代替キー・オーバーフロー・ブロックにあった代替キー・エントリーが、それよりも前の代替キー・オーバーフロー・ブロックまたは代替キー・プライマリー・ブロックに格納されてしまう場合には、当該代替キー・オーバーフロー・ブロックが不要となるが、この場合は、当該代替キー・オーバーフロー・ブロックを未使用ブロックとしてもよいし、そのまま接続しておいてもよい。接続しておく場合の、当該代替キー・オーバーフロー・ブロックの代替キー値の最小値と最大値は、直前の代替キー・プライマリー・ブロックまたは代替キー・オーバーフロー・ブロックの代替キー値と、次の、代替キー・ロケーション・テーブル・エントリーの代替キーの最小値と矛盾しない範囲で、最小値と最大値を同一にしておくことが好ましい。

【０１５４】

この代替キー・エントリーの追加と削除によって、代替キー・ロケーション・テーブルが変更される場合は、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合と、削除される代替キー値が当該ブロック中で最小または最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。更新後の代替キー・ロケーション・テーブル・エントリーの内容（Ａログ）と、どの代替キー・ロケーション・テーブル・エントリーであるかの識別情報（ブロック番号）が情報である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同

様に実施する。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加・削除が完了したら、これまでの排他を解除する。

【０１５５】

[代替キーによるレコードの更新：アクセラレーター・システムの場合]

アクセラレーター・システム１上で行う場合には、まず、代替キー・ロケーション・テーブルＡＬ１をバイナリー・サーチして、更新するレコードの代替キー値を持つ代替キー・ロケーション・テーブル・エントリーを見つけ出す。

代替キー・ロケーション・テーブル・エントリーを検索した後、その代替キー・ロケーション・テーブル・エントリーが指しているプライマリー・システム０上の代替キー・ブロック内を検索し、代替キー・エントリーを検索する。代替キー・エントリーが検索できたら、再度、アクセラレーター・システム１上で、ロケーション・テーブルＬ１を使用して、バイナリー・サーチを行い、レコードが格納されているブロックを検索する。

この動作は、代替キーによる検索で述べた方法と同様である。

代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。更新を代替キーで実行する場合には、同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、更新を行うか否かの選択を行うことになる。ここでは、複数のレコードを順次更新するものとする。

【０１５６】

検索したロケーション・テーブル・エントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にあるブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「特願２００２－２６４２８３号　データ及びデータベースの無停止

自動再編成システム、並びに、データ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式を使用することが、排他の高速化のために好ましい。

ブロック「n」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。ブロック内にターゲット・キー値を持つレコードが存在しない場合は有り得ない。

このようにして、レコードの更新が行われるが、主キー値の更新は許されないので、ロケーション・テーブルＬ０の変更は行われない。主キー値の更新を行いたい場合は、当該主キー値を持ったレコードを一旦削除し、新たな主キー値を持ったレコードを追加すると言う方法を採る。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

【０１５７】

[代替キーによるレコードの更新に伴う代替キー・エントリーの追加、削除：プライマリー・システムの場合]

レコードが更新されると、それに伴って代替キー・エントリーの更新、実際には追加と削除が行われる場合がある。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード更新の場合は、必ず代替キー・エントリーの更新が行われるわけではなく、レコードの項目（フィールド）で代替キーに指定した項目が更新された場合に、代替キー・エントリーの更新が必要となる。

代替キーが３種類ある場合には、レコード更新によって０種類から３種類の代替キー・エントリーの更新が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは省略する。

ここでは、すべての代替キー・エントリーが更新される場合を例にとって説明する。新しい代替キー・エントリーがＡｅｎ、Ｂｅｎ、Ｃｅｎの３つ作成され、これらの代替キー・エントリーの追加が必要になる。逆に、当該レコードのそれまでの代替キー・エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏは不要となり、削除が必要

になる。

この代替キー・エントリーの追加・削除は、「主キーによるレコード更新による、代替キー・エントリーの更新」と方式が同様であるので、説明は省略する。

【０１５８】

[主キーによるレコードの削除：アクセラレーター・システムの場合]

アクセラレーター・システム１上で行う場合には、まず、ロケーション・テーブルＬ１をバイナリー・サーチして、削除するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する。ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行い、ターゲット・キー値が当該ロケーション・テーブル・エントリーの最小値と最大値の間である場合には、当該ロケーション・テーブル・エントリーが管理しているブロック（当該プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも含む）にレコードが格納されてい得ることになる。

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の何れか一方を保持している場合には、前後のロケーション・テーブル・エントリーの主キー値（最小値または最大値）と比較して、対象ブロックを決定する。

【０１５９】

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「特願２００２－２６４２８３号　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式を使用する

ことが、排他の高速化のために好ましい。

【０１６０】

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。そのレコードを削除処理することになる。ブロック内にターゲット・キー値を持つレコードが存在しない場合は、当該レコードは存在しないことになる。

このようにして、レコードの削除が行われる。レコードを削除すると、削除したレコードが占めていた領域が空くことになる。このままでは、その後の主キーによる検索に支障を来たすので、削除するレコードより上位アドレスにあるレコードを削除対象レコードが占めていた領域の分だけアドレスの下位方向に動かしておく必要がある。また、下位方向に移動することによって、オーバーフロー・ブロックにあったレコードが、それよりも前のオーバーフロー・ブロックまたはプライマリー・ブロックに格納されてしまう場合には、オーバーフロー・ブロックが不要となるが、この場合は、当該オーバーフロー・ブロックを未使用ブロックとしてもよいし、そのまま接続しておいてもよい。接続しておく場合の、当該ブロックの主キー値の最小値と最大値は、直前のプライマリー・ブロックまたはオーバーフロー・ブロックの主キー値と、次の、ロケーション・テーブル・エントリーの主キーの最小値と矛盾しない範囲で、最小値と最大値を同一にしておくことが好ましい。

【０１６１】

このレコードの削除によって、ロケーション・テーブルが変更される場合は、削除される主キー値が当該ブロック中で最小または最大のキー値を持つ場合で且つ、代替キー・エントリーに主キー値の最小値または最大値を持っている場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

【０１６２】

[主キーによるレコードの削除に伴う代替キー・エントリーの削除：プライマリ

ー・システムの場合]

ところが、レコードが削除されると、それに伴って代替キー・エントリーの更新、実際には削除が行われる。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード削除の場合は、必ず代替キー・エントリーの更新が行われる。

代替キーが３種類ある場合には、レコード削除によって３種類の代替キー・エントリーの削除が行われる。

これまでの代替キー・エントリーＡｅ、Ｂｅ、Ｃｅの３つが削除対象となる。代替キー・エントリーの削除は、代替キー・エントリーの追加と削除の中の削除に関するところで述べた説明と同様であるので、ここでは省略する。

【０１６３】

この代替キー・エントリーの追加と削除によって、代替キー・ロケーション・テーブルが変更される場合は、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合と、削除される代替キー値が当該ブロック中で最小または最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

【０１６４】

それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。更新後の代替キー・ロケーション・テーブル・エントリーの内容（Ａログ）と、どの代替キー・ロケーション・テーブル・エントリーであるかの識別情報（ブロック番号）が情報である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同様に実施する。

代替キー・テーブルA，B，Cへの代替キー・エントリーの追加・削除が完了したら、これまでの排他を解除する。

【0165】

[代替キーによるレコードの削除：アクセラレーター・システムの場合]

アクセラレーター・システム1上で行う場合には、まず、代替キー・ロケーション・テーブルAL1をバイナリー・サーチして、削除するレコードの代替キー値を持つ代替キー・ロケーション・テーブル・エントリーを検索する。

代替キー・ロケーション・テーブル・エントリーを検索した後、プライマリー・システム上の代替キー・ブロック内を検索し、代替キー・エントリーを見つけ出す。代替キー・エントリーが検索できたら、ロケーション・テーブルL1を使用して、バイナリー・サーチを行い、レコードが格納されているブロックを検索する。

この動作は、代替キーによるレコードの検索で述べた方法と同様である。

代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。削除を代替キーで実行する場合には、同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、削除を行うか否かの選択を行うことになる。ここでは、複数のレコードを順次削除するものとする。

【0166】

検索したロケーション・テーブル・エントリーが、ブロック番号「n」を指していたとする。ブロックはプライマリー・システム0に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム0にあるブロック「n」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム0のロケーション・テーブルL0の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「データ及びデータベースの無停止自動再編成システム並びにデータ及びデータベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式を使用することが、排他の高速化のために好ましい

。

【0167】

ブロック「n」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。ブロック内にターゲット・キー値を持つレコードが存在しない場合は有り得ない。

この後、「主キーによるレコードの削除」の場合と同様に、ブロック内で削除レコードが占めていた領域を詰め、必要に応じてロケーション・テーブル・エントリーの更新を行い、代替キー・エントリーの削除を行う。

このようにして、レコードの削除を行う。

一連の動作が完了したら、排他解除を行う。

以上で、アクセラレーター・システム内でのデータ（レコード）の検索、追加、更新、削除に伴うプライマリー・システム内での動作と、それに伴う、アクセラレーター・システムの動作に関して述べた。

【0168】

[アクセラレーター・システムを複数作成する場合]

以上で、アクセラレーター・システム１が存在する場合に関して述べたが、アクセラレーター・システムを複数作成することが可能である。アクセラレーター・システム２、３、４、・・・・、ｎが存在する場合の動作は、単一のアクセラレーター・システムの場合と殆ど同様である。違いは、プライマリー・システムでロケーション・テーブルまたは代替キー・ロケーション・テーブルに変更が発生した場合に、アクセラレーター・システム１にのみ情報を送信し、情報の更新完了を待つのではなく、すべてのアクセラレーター・システム２、３、・・・・、ｎに情報を送信し、すべてのアクセラレーター・システムでの情報の更新完了を待つ、ということである。

【0169】

ロケーション・テーブル及び代替キー・ロケーション・テーブルの更新は、プライマリー・システム上で最初に発生するが、そのきっかけとなるのは、各アクセラレーター・システムからの、データ追加、更新、削除要求である。アクセラレーター・システム１，２、３、・・・・、ｎからのデータ追加、更新、削除要

求で、プライマリー・システムのロケーション・テーブル及び代替キー・ロケーション・テーブルの更新が発生すると、その更新情報を、すべてのアクセラレーター・システムに送信し、同期更新を行う。

また、ロケーション・テーブル、または、代替キー・ロケーション・テーブルに変更があった場合にのみ変更情報を送信するので、従来の方法のように、完全なミラー・コピーを持つ方法に比較して、データ送信量が大幅に少なく、また、アクセラレーター・システムで必要な格納領域が、プライマリー・システムに比較して大幅に少なくて済むという利点がある。

更に、各アクセラレーター・システムでエントリーに対する検索を独立して行うので、システムの負荷が分散でき、従来の方法に比較して、大幅なスケーラビリティの向上が行える。

【０１７０】

[バックアップ・リカバリー方式との連携]

次に、「データバックアップ・リカバリー方式」との同時使用が可能であることを説明する。

バックアップ・リカバリーは同期密結合方式と、非同期疎結合方式がある。同期密結合方式のバックアップ・リカバリーと、アクセラレーター・システムにおける、ロケーション・テーブルおよび／または代替キー・ロケーション・テーブルの更新は、同期を取って実行する。しかしながら、バックアップ・リカバリーは、ロケーション・テーブルおよび／または代替キー・ロケーション・テーブルの更新よりも、ブロックおよび／または代替キー・ブロック（何れもオーバーフロー・ブロックを含む）の更新機会の方がはるかに多いので、バックアップ・リカバリーにおけるセカンダリー・システムの更新の都度、アクセラレーター・システムの更新が必要になるものでは無い。

バックアップ・リカバリーが非同期疎結合方式の場合には、アクセラレーター・システムのみ、同期密結合方式で実行する。

【０１７１】

[再編成システムとの連携]

次に、「データ及びデータベースの無停止自動再編成システム、並びに、デー

タ及びデータベース」との同時使用が可能であることを説明する。

プライマリー・システムで再編成を行う場合には、同時にアクセラレーター・システムにも、現用のロケーション・テーブルに対して、新規のロケーション・テーブルを用意して、ロケーション・テーブルとブロックの再編成を実行する。プライマリー・システムとアクセラレーター・システムの再編成は同期をとって実行する。

アクセラレーター・システムでも、再編成ポインターを保持し、プライマリー・システムの再編成ポインターと同期させる。同期させるというのは、再編成ポインターが指しているロケーション・テーブル・エントリーの位置が、プライマリー・システムとアクセラレーター・システムで同じにするということである。検索等のターゲット・キー値が再編成ポインターが指しているロケーション・テーブルの主キー値の最小値よりより小さい場合には、新規のロケーション・テーブルを使用し、検索等のターゲット・キー値が再編成ポインターが指しているロケーション・テーブルの主キー値の最小値以上である場合には、現用のロケーション・テーブルを使用して、バイナリー・サーチを実行し、目的のレコードを検索する。

【０１７２】

同様に、プライマリー・システムで代替キー・テーブルの再編成を行う場合には、同時にアクセラレーター・システムにも、現用の代替キー・ロケーション・テーブルに対して、新規の代替キー・ロケーション・テーブルを用意して、代替キー・ロケーション・テーブルと代替キー・ブロックの再編成を実行する。プライマリー・システムとアクセラレーター・システムの再編成は同期をとって実行する。

アクセラレーター・システムでも、再編成ポインターを保持し、プライマリー・システムの再編成ポインターと同期させる。同期させるというのは、再編成ポインターが指しているロケーション・テーブル・エントリーの位置が、プライマリー・システムとアクセラレーター・システムで同じにするということである。

検索等のターゲット・キー値が再編成ポインターが指している代替キー・ロケーション・テーブルの代替キー値の最小値よりより小さい場合には、新規の代替

キー・ロケーション・テーブルを使用し、検索等のターゲット・キー値が再編成ポインターが指している代替キー・ロケーション・テーブルの主キー値の最小値以上である場合には、現用の代替キー・ロケーション・テーブルを使用して、バイナリー・サーチを実行し、目的の代替キー・エントリーを検索する。

代替キー・エントリーが見つかったら、そのエントリーが持っている情報により、ロケーション・テーブルからブロックを探し、レコードの検索を行う。

このように行うことで、再編成システムとの連携が可能である。

【０１７３】

[対称システム]

プライマリー・システムは、１つのロケーション・テーブルとｇ種類の代替キー・テーブルを持つ構造をしている場合に、この（１＋ｇ）個を組として、アクセラレーター・システムをｎ個作成する構成（対称アクセラレーター・システム）とすることが可能である。

【０１７４】

[非対称システム]

この方法のほかに非対称アクセラレーター・システムを構成することも可能である。これは、プライマリー・システムのロケーション・テーブルと、代替キー・テーブルＡ、Ｂ、Ｃ、・・・に対して、アクセスの頻度により、アクセラレーターの数を決定する方法である。

例えば、プライマリー・システムが、ロケーション・テーブルと代替キー・テーブルＡ、Ｂ、Ｃの３種類の代替キー・テーブルを持っている場合について説明する。

アクセラレーター・システム１は、ロケーション・テーブル１と代替キー・テーブル１Ａ、１Ｂ、１Ｃを保有し、プライマリー・システムと同様の構成である。アクセラレーター・システム２はロケーション・テーブル２と、代替キー・テーブル２Ｂ、２Ｃを保有し、プライマリー・システムと比較すると、代替キー・テーブル２Ａを保有しない形態となる。

同様に、アクセラレーター・システム３はロケーション・テーブル３と代替キー・テーブル３Ｃを保有する。

これは主キーと代替キー各々の使用頻度が分かっていて、その頻度に大きな違いがある場合に有効である。対称アクセラレーター・システムの場合は、あまり使用されないテーブルも保有することになり、格納領域の無駄が発生するからである。

【０１７５】

また、アクセラレーターに対するアクセスの統計をとるようにし、主キーや代替キー毎のアクセスが変動する場合には、各アクセラレーターの個数を変更することが好ましい。また、次のように個数変更を自動化することも可能である。

自動化する方式を以下に述べる。

まず、アクセラレーター・システムのキー毎に統計をとる。そして各キー毎のアクセス量を比較する。一定以上のアクセスの場合、非対称アクセラレーターの追加を行う。

【０１７６】

例として下記のような、非対称アクセラレーター・システムが設定されているとする。

【０１７７】

非対称アクセラレーター・システム１、
ロケーション・テーブル１、
代替キー・テーブル１Ａ、
代替キー・テーブル１Ｂ、
代替キー・テーブル１Ｃ、

【０１７８】

非対称アクセラレーター・システム２、
ロケーション・テーブル２、
代替キーテーブル２Ｂ、
代替キーテーブル２Ｃ、

【０１７９】

非対称アクセラレーター・システム３、
ロケーション・テーブル３、

代替キー・テーブル３Ｃ、

【０１８０】

という構成だった場合、代替キー・テーブルＢのアクセスが増加して、追加が必要だとすると、非対称アクセラレーター・システム２に対して、代替キー・テーブル２Ａを追加する。非対称アクセラレーター・システム３に追加してもよいが、本事例では、アクセラレーター・システムのうち、番号の若いシステムに追加することとする。追加の方法は、以下に述べるとおりである。

【０１８１】

追加するということを、プライマリー・システムで決定する。決定後、まず、アクセラレーター・システム２に対して、代替キーテーブル２Ａ用の領域確保を命ずる。アクセラレーター・システム２では、その情報に基づいて領域の確保を行う。領域の確保が完了したら、プライマリー・システムに対して、完了した旨を送信する。その後、プライマリー・システムは、アクセラレーター・システム２に対して、代替キーテーブルＡのすべてのエントリーをＡログとして送信する。送信が完了するまでの間に発生する、データ更新に伴うＡログも、同様に送信する。

アクセラレーター・システム２では、Ａログの内容に基づいて、代替キーテーブル２Ａの該当するエントリーを順次更新する。Ａログの更新がすべて完了した時点で同期が完了したことになり、代替キー・テーブル２Ａを使用することが可能となる。

以上では、プライマリー・システムから、情報を送信する例を示したが、プライマリー・システムの負荷が大きくなりすぎる場合には、アクセラレーター・システム２以外のアクセラレーター・システムから送信するようにしても良い。

【０１８２】

以上の方式を適用する際に、高速性を確保するためには、各々のアクセラレーターに対してプロセッサーを割り当てることが好ましい。更には、アクセラレーター・システムのロケーション・テーブルと代替キー・テーブルの各々に対して、プロセッサーを割り当てる方式は、より高速性を高める上で好ましい。

アクセラレーター・システムは、プライマリー・システムと同一のハードウエ

アに内蔵する形式が可能であるが、その他に、アクセラレーター・システム毎に、プライマリー・システムとハードウエアを分離することも可能である。

このようにしてアクセラレーター・システムの数を増やしていくと、テーブルのアクセス数が増加するために、検索等の対象となるブロックのアクセスが増加して、ブロックをアクセスするためのプロセッサーの能力が限界になる可能性があるが、この場合には、ブロックを複数個のバンクに分離し、その各々にプロセッサーを割り当てることによって、負荷分散を図ることが可能である。

【０１８３】

[格納領域の優位性]

「データバックアップ・リカバリー方式」および、「データバックアップ・リカバリー方式」で示した、セカンダリー・システムを参照用として用いる場合に比較して、格納領域が小さくて済む優位性に関して述べる。

ファイル（ブロックの集合）に対して、ロケーション・テーブルと代替キー・テーブルの大きさは、レコードの長さ、キーの長さ、ブロックの大きさ、ブロックへの格納率など、定める条件によって異なるため、厳密に述べることは不可能であるが、ファイルに対して、ロケーション・テーブルの大きさは１００分の１程度、代替キー・テーブル１種類の大きさは２０分の１程度であると想定できる。

このため、「データバックアップ・リカバリー方式」および、「データバックアップ・リカバリー方式」で示した、セカンダリー・システムを参照用として用いる場合に比較して、代替キー・テーブルが５種類ある場合で、５つの対称アクセラレーター・システムを採用した場合とすると、以下のようになり、約５分の１の格納容量で済むことになる。アクセラレーター・システムの数を増やせば同じ比率で増加することになる。また、非対称アクセラレーター・システムを採用した場合には、更に比率が高まることになる。

【０１８４】

プライマリー・システムの大きさ（ブッロクの大きさを１００とした場合の全体の大きさ）：

１＋１／１００＋５×（１／２０）＝１２６／１００

【０１８５】

５つのセカンダリー・システムを用意した場合：

（１２６／１００）×５＝６３０／１００

【０１８６】

５つのアクセラレーターを用意した場合：

（２６／１００）×５＝１３０／１００

【０１８７】

つまり、

６３０：１３０算≒５：１

となる。

【０１８８】

「データ格納検索方式」では、ロケーション・テーブルの構造として、ロケーション・テーブルの番号（＝プライマリー・ブロックの番号）と、ブロックの位置を保持し、ブロック中のレコードの主キー値の最小と最大のものを、両方またはどちらか一方を保持することも可能としてある。主キー値をロケーション・テーブルで保持しない場合は、ロケーション・テーブルに対する更新が減少するが、バイナリー・サーチでブロックを探す都度、ブロックを読まないと目的のブロックであるか否かの判定ができないため、ブロックに対する負荷が増加するので、本発明に適用する場合には、ロケーション・テーブルにキー値を保持する方式が好ましい。キー値は最小と最大の両方を保持している場合には、そのエントリーを読むだけで目的のブロックであるか否かの判定が可能である。どちらか一方を保持している場合には、そのエントリーの後、または、前のエントリーを読むことにより、解決が行える。

【０１８９】

ロケーション・テーブルのコピー方法は、「データバックアップ・リカバリー方式」および、「データバックアップ・リカバリー方式」で記述してある方式を用いるのが効率的である。これは、予め、プライマリー・システム上にあるロケーション・テーブルのエントリーと同数のエントリーを持つセカンダリー・ロケーション・テーブルを用意しておき、プラーマリー・ロケーション・テーブルの

エントリーに変更が発生したら、セカンダリー・ロケーション・テーブルを持つセカンダリー・システムにその変更を通知し、セカンダリー・ロケーション・テーブルを変更するというものである。このセカンダリー・システムに対して行う操作を、アクセラレーター・システムに対して行うことで実現される。

代替キー・テーブルは、ブロックと同様の構成であるので、ブロックに対する更新をセカンダリー・システムに適応するのと同様のロジックで、アクセラレーター・システム・代替キー・テーブルに対して更新を行う。

【０１９０】

ロケーション・テーブルは、ロケーション・テーブルのエントリーが保持しているブロックの位置が変更されることはなく、ブロックに含まれるレコードの最小値、最大値の双方または何れか一方が変更された場合に変更が発生するため、旧来の方法によるインデックスに比較して、変更の発生ははるかに少ない。旧来の方法では、ブロックに相当するものの分割が発生して、ロケーション・テーブルのエントリーに相当するものの数が増減するため変更が多いのである。また、旧来の方法でのインデックスでは、階層構造をとっているため、上位の階層の変更が発生するケースが頻繁にあり、これも、旧来の方法でインデックスの変更が頻繁に発生する原因となっていた。

本発明を適用すると、アクセラレーター・システムが複数存在することになり、数が増加すると、各々に対してプライマリー・システムから変更を通知し、その結果を待つための負荷が増大する。これを軽減する方法として、一斉同報方式を用いて、セカンダリー・システムに対して変更を通知する方式が有利である。

セカンダリー・システムでセカンダリー・ロケーション・テーブルの更新が完了した通知は各々のセカンダリー・システムから個別にプライマリー・システムに対して送信する。

【０１９１】

各アクセラレーター・システムに対して、どのように要求を割振るかに関して述べる。このデータ格納・検索システムに対する要求は、ＳＱＬのような形式の他、Ｒｅａｄ／Ｗｒｉｔｅインターフェイスも可能であるが、それらは、最終的には、次のような情報となる。

【0192】

1．対象となるキーの種類（主キー、代替キーA、B、C・・・）

2．アクセスの種類（Ｒｅａｄ、Ｗｒｉｔｅ、Ｒｅｒｉｔｅ、Ｄｅｌｅｔｅ、Ｉｎｓｅｒｅｔ　など）

3．要求キー値

4．データ内容（Ｉｎｓｅｒｔ　の場合）

【0193】

これらの情報が一組となって、データ格納・検索システムに対しての処理要求となる。振り分けには、図3、図4、図8を用いて説明する。

図3に示した図は、処理要求を振り分けるのに、ロード・バランサーを使用する方式を示したものである。ロード・バランサーとは、複数のサーバーが、見かけ上、同じＩＰアドレスを保有しているかのように見せかけ、多数の端末からの接続を、複数のサーバーに振り分けるためのものである。この場合、処理サーバー（一般的には、アプリケーション・サーバーと呼ばれる場合が多い）と、プライマリー・システム、または、アクセラレーターは固定的に接続される。

【0194】

図4に示したシステムは、処理要求振り分けシステムと呼ぶことにする。また、図8は処理要求振り分けシステムの動作を詳細に説明するためのものである。処理要求振り分けシステムは、ロード・バランサーを使用する場合のように、固定的な接続とせず、処理要求毎に、プライマリー・システムとアクセラレーターを選択する方式である。

図3および図4では、インターネットを使用して、端末から処理要求が送信される図となっているが、インターネットを使用せず、専用線やＬＡＮを用いる構成でも適用可能である。

【0195】

図8中の処理要求には、一連の番号と、キーの種類のみを抜き出して示してある。その他の情報は、これに付属しているものとする。処理要求は、通常用いられているようにキュー構造をするのが好ましい。

図8には処理システム・テーブルが4つ記載されている。各々は、ロケーショ

ン・テーブル用の処理システム・テーブルＬ、代替キー・テーブルＡ用の処理システム・テーブルＡ、代替キー・テーブルＢ用の処理システム・テーブルＢ、代替キー・テーブルＣ用の処理システム・テーブルＣ、となっている。

【０１９６】

各処理システム・テーブルのエントリーは、各アクセラレーター・システムに対応しており、図８の場合は、ロケーション・テーブルのアクセラレーター・システムが１９個、代替キー・テーブルＡのアクセラレーター・システムが１４個、代替キー・テーブルＢのアクセラレーター・システムが１１個、代替キー・テーブルＣのアクセラレーター・システムが１７個、存在する場合を示している。

各処理システム・テーブルのエントリーは、処理システム番号と処理中フラッグから構成されている。処理システム番号は、ゼロはプライマリー・システムを、１からの正の数字は、アクセラレーター・システムを示している。処理中フラッグは、そのシステムが処理要求を処理中であるか、非処理状態であるかを識別するためのものである。

【０１９７】

図８中に白抜き矢印があるが、これは、各処理システム・テーブルの処理を、プライマリー・システムまたはアクセラレーター・システムのうちのどれが行っているか、システムの最新番号を示している。ゼロはプライマリー・システムを、１からの正の数字は、アクセラレーター・システムを示している。この矢印を、処理システム・ポインターと呼ぶ。この処理システム・ポインターは、テーブル（キー）の種類毎に保持する必要があるため、ロケーション・テーブル用の処理システム・ポインターＬ、代替キー・テーブルＡ用の処理システム・ポインターＡ、代替キー・テーブルＢ用の処理システム・ポインターＢ、代替キー・テーブルＣ用の処理システム・ポインターＣ、などとなっている。

処理システム・ポインターは、初期は、各処理システム・テーブルの先頭を指しているが、処理要求が割り当てられる度に、次のエントリーを指すようにする。各処理システム・テーブルの最終エントリーの次を指す場合には、先頭に戻ることとする。

このようにして、各アクセラレーターに、処理要求が均等に割り当てられるよ

うにしている。

図８は処理要求割り当て後を示しているため、各、処理システム・ポインターの位置が、移動した後の状態を示している。

【０１９８】

処理要求の、１３８はロケーション・テーブルに対する処理要求である。この処理要求が来た時点では、処理中の処理要求が無かったものとする。処理要求１３８は、図中の処理システム・テーブルＬ（ロケーション・テーブルと書かれた表）を参照する。処理中の処理要求が無い時点で発生した処理要求であるので、処理要求振り分けシステムでは、処理システム・テーブルＬの処理システム番号：０のエントリーに対して、処理中のフラッグを立てる。次に、処理システム・ポインターを処理システム番号：１に移動する。次に、処理要求１３８にシステム番号：０を割り振り、処理要求１３８をプライマリー・システムに送信する。

プライマリー・システムでは、処理要求１３８に基づき、ロケーション・テーブルに対する処理を行い、処理が完了したら、振り分けシステムに対して、処理要求１３８が完了したことを通知する。振り分けシステムでは、プライマリー・システムからの完了通知であるので、処理システム・テーブルＬの処理システム番号：０の処理中のフラッグを、非処理にする。

【０１９９】

処理要求：１３９は代替キー・テーブルＡに対する要求であるので、処理システム・テーブルＡを参照する。処理システム・ポインターＡは、処理システム・テーブルの先頭を指しているため、処理要求１３９は、プライマリー・システムに割り当てる。

この割り当て作業は、処理要求：１３８の完了を待つ必要はなく、処理要求：１３８の割り当てを完了したら、即座に実行することが可能であるし、そうすることにより、高速な処理が行える。

同様に、処理要求：１４０は代替キー・テーブルＢに対する処理要求であるので、処理システム・テーブルＢを参照する。処理システム・ポインターＢは、処理システム・テーブルの先頭を指しているため、処理要求１４０は、プライマリー・システムに割り当てる。

その後の、動作は、処理要求：１３８と同様である。処理システム・ポインターは、処理システム・テーブルの最後までポイントしたら、先頭に戻るように設計しておく。先頭に戻った場合に、処理システム・ポインターが指している処理システム番号が処理中の場合は、次の要求を待ちにするか、または、次々とシステムを検索し、処理中で無いシステムを見つけて、要求をそのシステムに割り当てるとともに、処理システム・ポインターをそこまで移動するという方法も可能である。後者の方法を採用する場合には、一時的にループする可能性が存在するが、処理システム・テーブルに割り当てられた処理要求の何れかが完了すれば、処理システム・テーブルの当該システム＃の処理中のフラッグが、非処理状態になるため、無限ループは発生しない。

【０２００】

【発明の効果】

１．データベース・システムに対して、大きなスケーラビリティを提供することが可能である。

２．スケーラビリティは、アクセラレーター・システムの数を順次増加させることにより段階的に拡張が可能である。

３．アクセラレーター・システムでは、プライマリー・システムが保有しているデータベースの大きさに比較し、小さな領域を確保すればよいので、従来のようにプライマリー・システムと全く同じ大きさのデータベースを確保する必要が無い。このため、従来の方法に比較して安価に構築可能である。

４．データに対する更新はプライマリー・システム上でのみ実施されるため、更新トランザクションをアクセラレーター・システムから実行可能である。

５．データに対する更新はプライマリー・システム上でのみ実施されるため、プライマリー・システムでの更新が、アクセラレーター・システムに対して遅延することが無い。

６．「データバックアップ・リカバリー方式」との連動が可能で、バックアップに対する不安がない。

７．「データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース」との連動が可能であるので、プライマリー・システムで再編成

を行う場合の制約がない。

【図面の簡単な説明】

【図１】プライマリー・システムの代表的な例である

【図２】プライマリー・システムとアクセラレーター・システムの代表的な例である

【図３】ロード・バランサーを使用した要求振り分けの例である

【図４】要求振り分けシステムを使用した要求振り分けの例である

【図５】ブロックにレコードを挿入する場合の例である

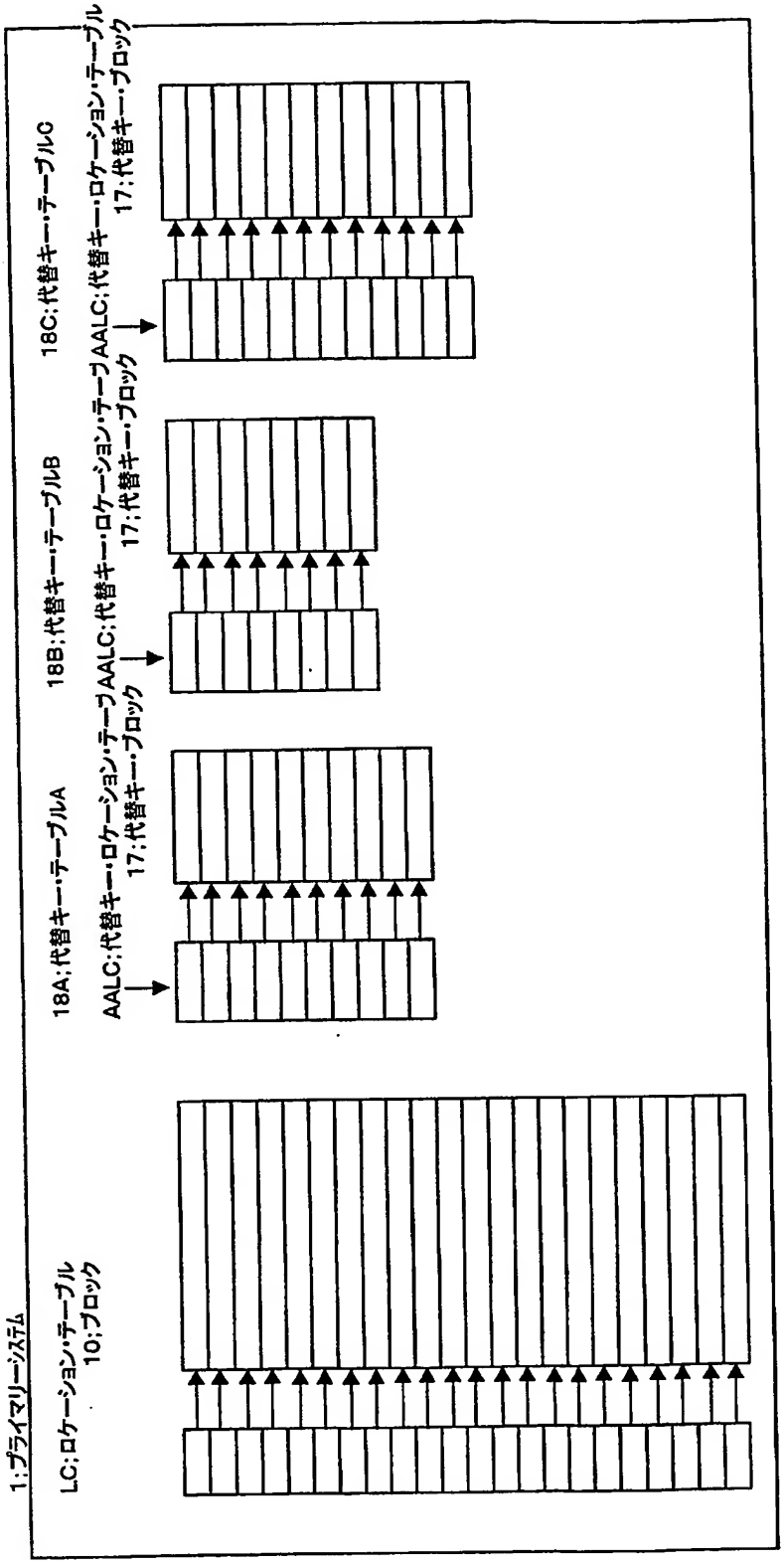
【図６】ブロックにレコードを挿入した場合に、オーバーフロー・ブロックが発生する場合の例である

【図７】代替キー・ブロックに代替キー・エントリーを挿入する場合の例である

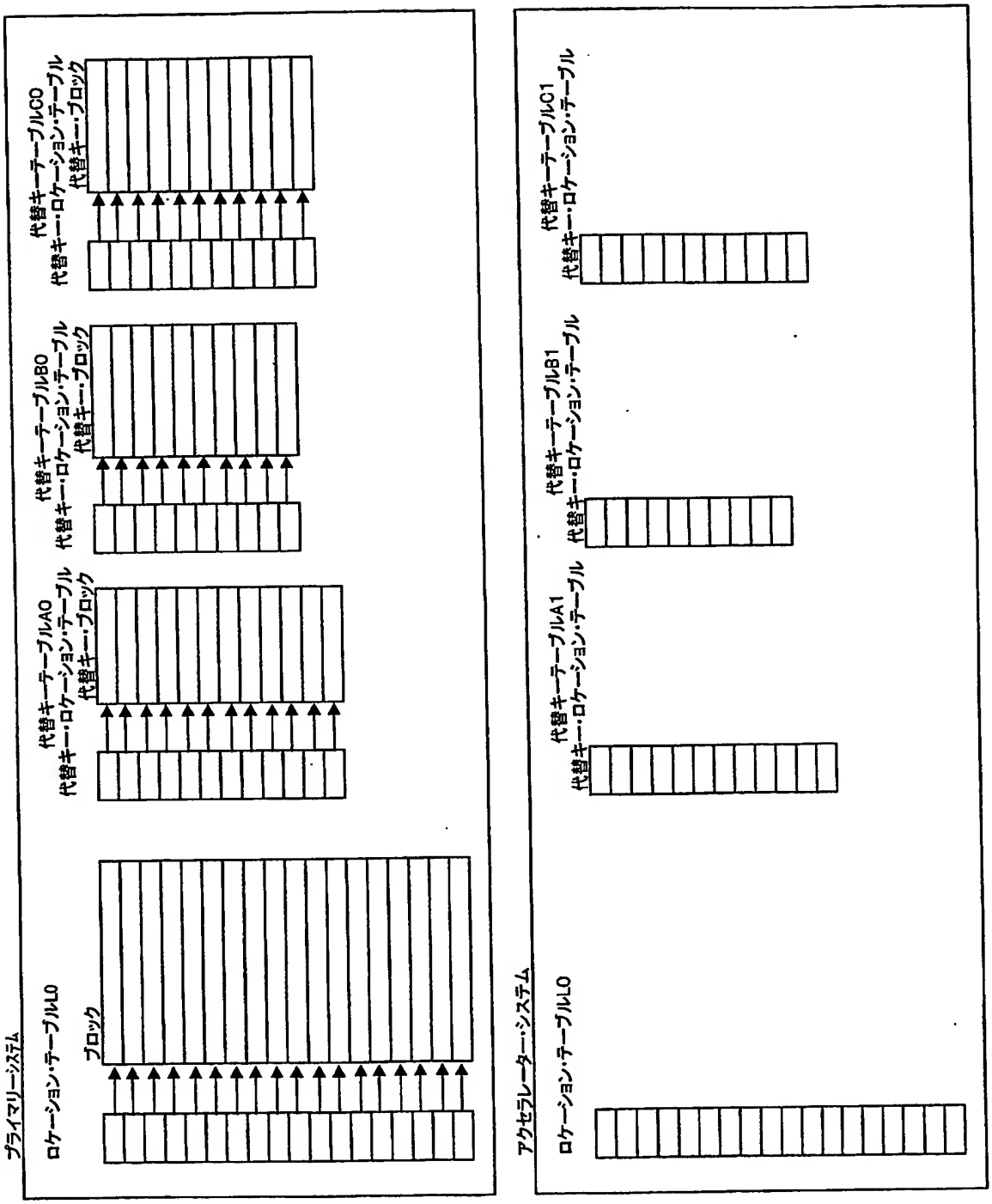
【図８】要求振り分けシステムを使用した場合の、要求振り分けの論理の例である

【書類名】　図面

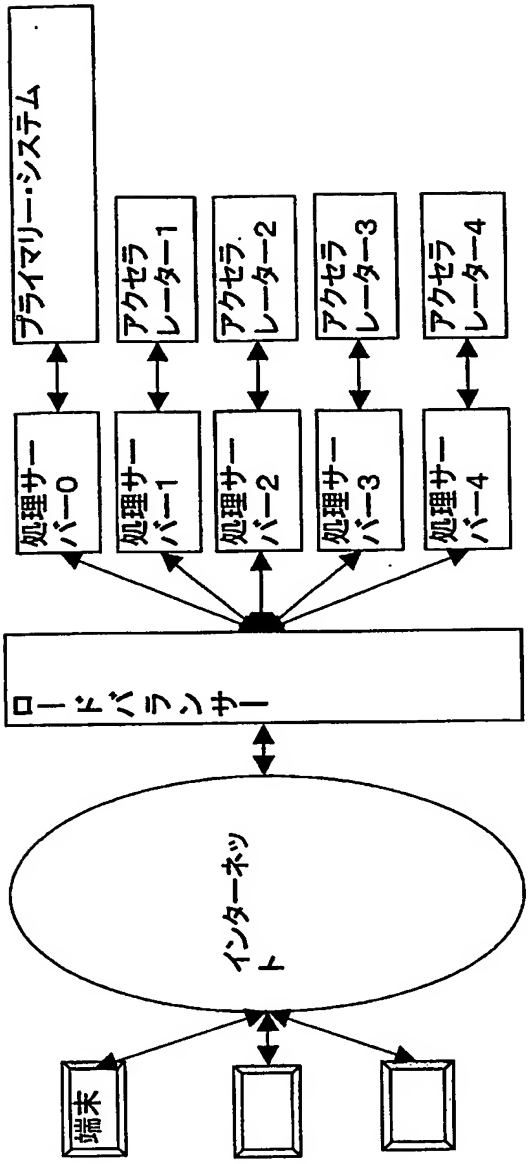
【図１】


1;プライマリーシステム
LC;ロケーション・テーブル
10;ブロック
AALC;代替キー・ロケーション・テーブル
17;代替キー・ブロック
18A;代替キー・テーブルA
AALC;代替キー・ロケーション・テーブル
17;代替キー・ブロック
18B;代替キー・テーブルB
AALC;代替キー・ロケーション・テーブル
17;代替キー・ブロック
18C;代替キー・テーブルC

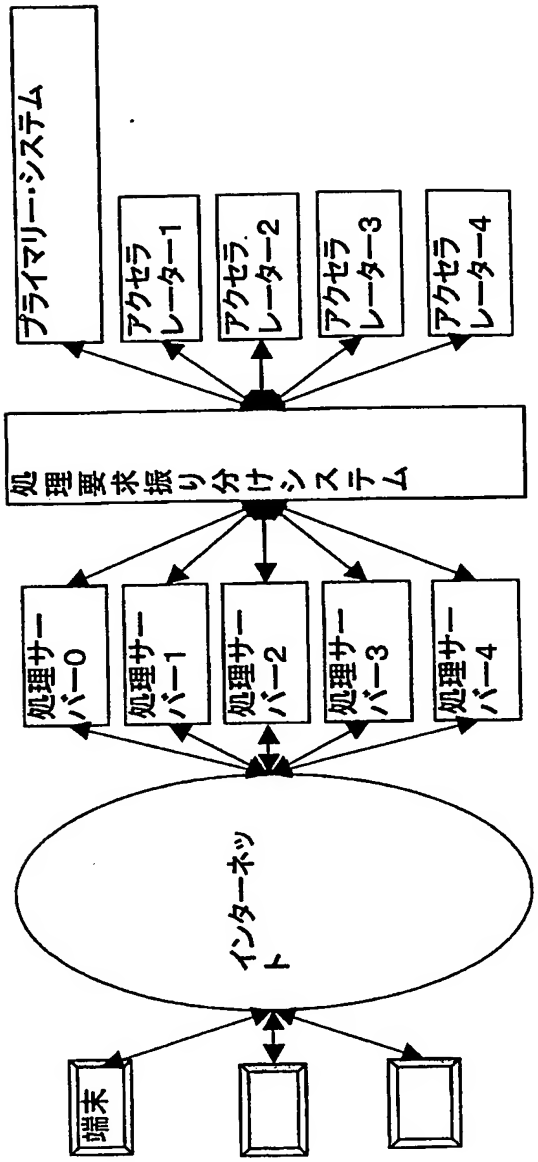
【図２】


プライマリーシステム
ロケーション・テーブルL0
ブロック
代替キー・ロケーション・テーブル
代替キー・ブロック
代替キーテーブルA0
代替キーテーブルB0
代替キーテーブルC0
アクセラレーター・システム
ロケーション・テーブルL0
代替キー・ロケーション・テーブル
代替キーテーブルA1
代替キーテーブルB1
代替キーテーブルC1

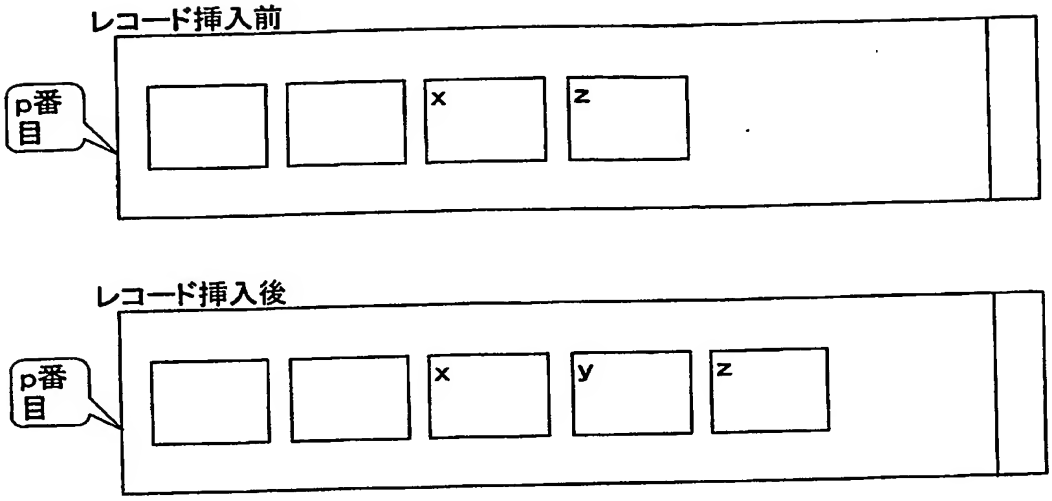
【図３】


プライマリー・システム
アクセラレーター1
アクセラレーター2
アクセラレーター3
アクセラレーター4
処理サーバー0
処理サーバー1
処理サーバー2
処理サーバー3
処理サーバー4
ロードバランサー
インターネット
端末

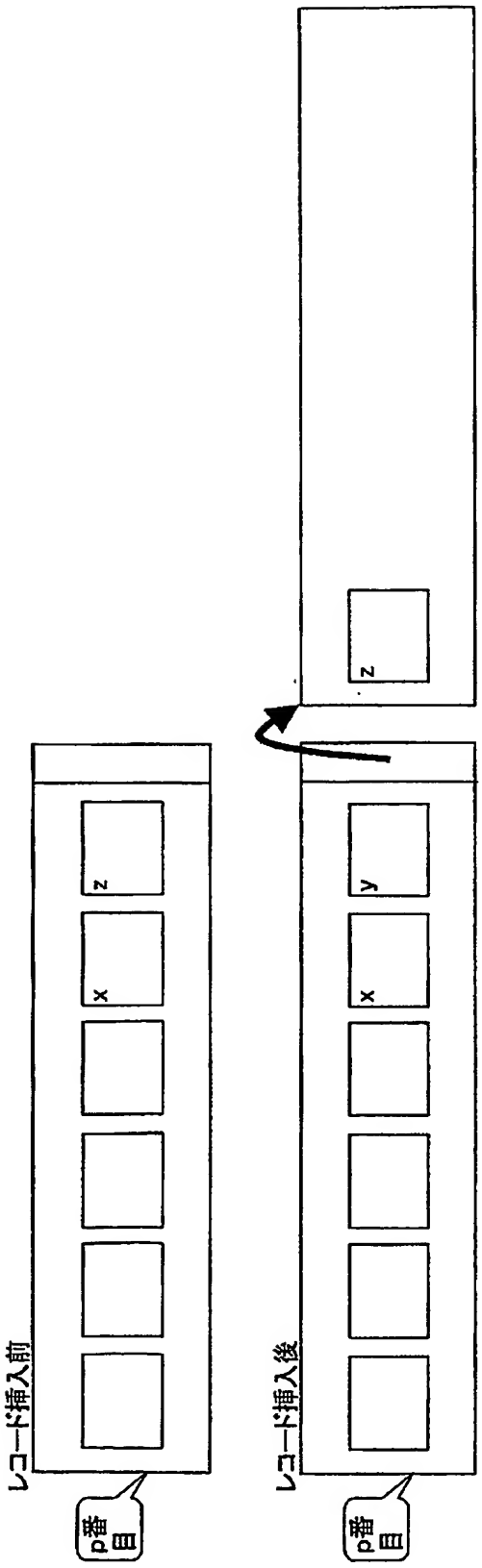
【図４】


プライマリー・システム
アクセラレーター1
アクセラレーター2
アクセラレーター3
アクセラレーター4
処理要求振り分けシステム
処理サーバー0
処理サーバー1
処理サーバー2
処理サーバー3
処理サーバー4
インターネット
端末

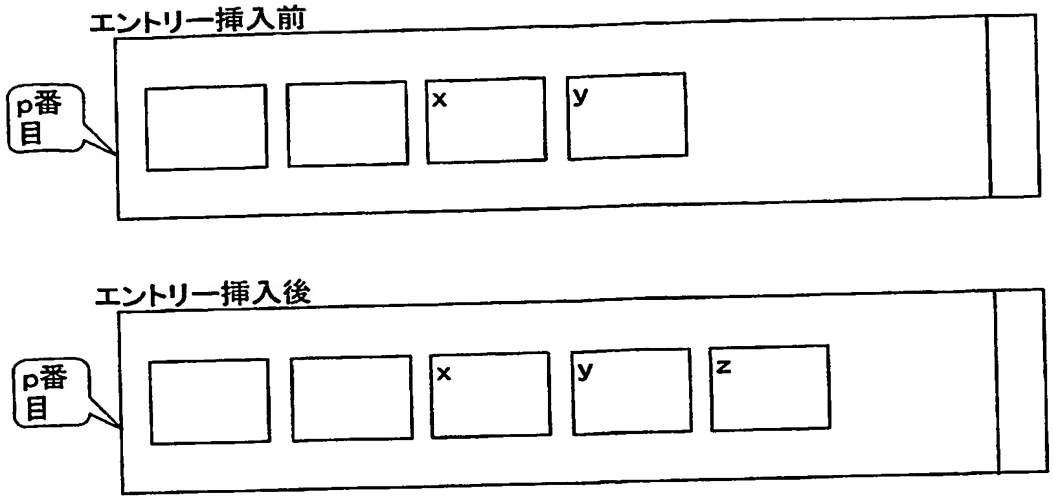
【図５】



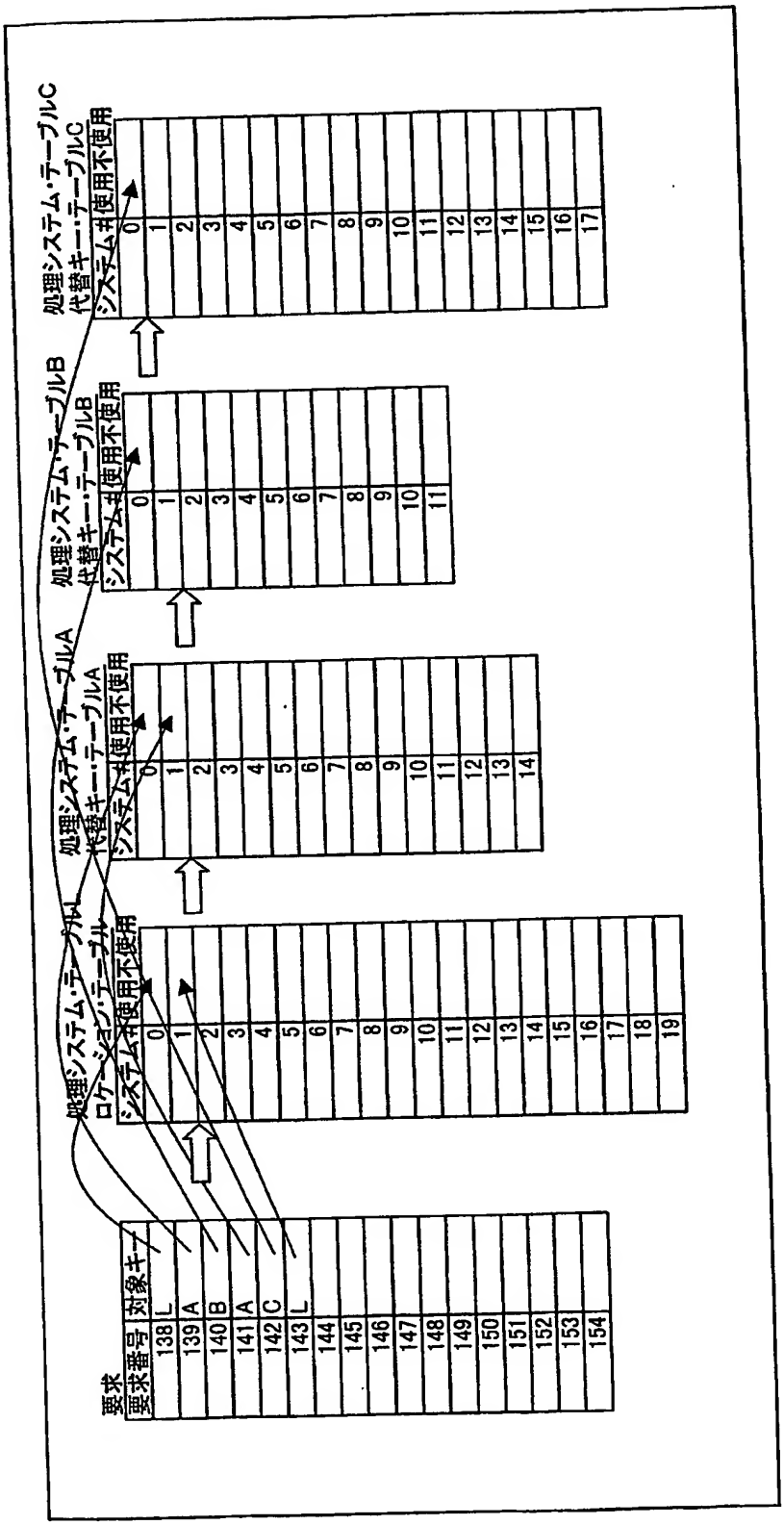
【図６】



【図７】



【図８】


処理システム・テーブルC
代替キー・テーブルC
システム#使用不使用
処理システム・テーブルB
代替キー・テーブルB
システム#使用不使用
処理システム・テーブルA
代替キー・テーブルA
システム#使用不使用
処理システム・テーブル1
ロケーション・テーブル
システム#使用不使用
要求
要求番号
対象キー
138 L
139 A
140 B
141 A
142 C
143 L
144
145
146
147
148
149
150
151
152
153
154

【書類名】　要約書

【要約】

【課題】　データベース処理におけるスケーラビリティの確保を、段階的に安価に可能とする一方で、数千倍程度の大きなスケーラビリティを提供すること。

【解決手段】　本方式は、データ更新が行なわれるプライマリー・システムに対して、主キー・インデックスに相当するロケーション・テーブルと代替キー・インデックスに相当する代替キー・ロケーション・テーブルのコピーをアクセラレーター・システム上に保有する。プライマリー・システムでロケーション・テーブル、代替キー・ロケーション・テーブルに変更があった場合は、アクセラレーター・システムのロケーション・テーブル、代替キー・ロケーション・テーブルの更新を同期して行う。データはプライマリー・システム上にのみ保有する。処理要求をアクセラレーター・システムに分散し、並列処理を行うことにより、処理能力の大きな拡張性を確保する。

【選択図】　図２

認定・付加情報

特許出願の番号	特願２００２－３０６２９６
受付番号	５０２０１５８３２８６
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１４年１０月２２日

＜認定情報・付加情報＞

【提出日】　平成14年10月21日

次頁無

特願２００２－３０６２９６

出願人履歴情報

識別番号　　[５９３０５０５９６]

１．変更年月日　　１９９３年　２月　４日
[変更理由]　　新規登録
住　所　　東京都港区北青山３丁目７番１号
氏　名　　アネックスシステムズ株式会社

特願２００２－３０６２９６

出　願　人　履　歴　情　報

識別番号　　[５９２１５９３２４]

１．変更年月日　　２００１年　８月　３日
[変更理由]　　住所変更
住　所　　東京都多摩市馬引沢２丁目１４番１４号　サンセットヒルズ２
氏　名　　玉津　雅晴